

DTIC FILE COPY



US Army Corps
of Engineers

(2)

AD-A196 529

*SIMULATED ENGINEER ASSESSMENT OF THE
COMMUNICATIONS ZONE MODEL (SEAC)
(Documentation and Users Manual)*

USAESC-R-88-3

ENGINEER

STUDIES

CENTER

DTIC
ELECTE
AUG 03 1988

S

CD

D

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official US Department of the Army position, policy, or decision unless so designated by other official documentation.

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

88 8 00 019 00

2

*SIMULATED ENGINEER ASSESSMENT OF THE
COMMUNICATIONS ZONE MODEL (SEAC)
(Documentation and Users Manual)*

USAESC-R-88-3

Prepared by
Engineer Studies Center
US Army Corps of Engineers

June 1988

DTIC
ELECTE
S **D**
AUG 03 1988
a D

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS N/A	
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) USAESC-R-88-3		7a. NAME OF MONITORING ORGANIZATION HQ ROK/US Combined Forces Command	
6a. NAME OF PERFORMING ORGANIZATION US Army Engineer Studies Center		7b. ADDRESS (City, State, and ZIP Code) APO San Francisco 96301	
6b. OFFICE SYMBOL (If applicable) CEESC		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION US Army Corps of Engineers		10. SOURCE OF FUNDING NUMBERS	
8b. OFFICE SYMBOL (If applicable) USACE		PROGRAM ELEMENT NO. PROJECT NO. TASK NO. WORK UNIT NO.	
8c. ADDRESS (City, State, and ZIP Code) 20 Massachusetts Avenue, NW Washington, DC 20314-1000		0 0 0 0	
11. TITLE (Include Security Classification) SIMULATED ENGINEER ASSESSMENT OF THE COMMUNICATIONS ZONE MODEL (SEAC) (Documentation and Users Manual) (U)			
12. PERSONAL AUTHOR(S) Halayko, Robert			
13a. TYPE OF REPORT Final		15. PAGE COUNT 312	
13b. TIME COVERED FROM 1987 TO 1988		14. DATE OF REPORT (Year, Month, Day) 8806	
16. SUPPLEMENTARY NOTATION AD Agency Accession No. DA305198			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD GROUP SUB-GROUP		Model; Simulation	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) SEAC is a computer model used to conduct a large class of engineering analyses for COMMZ operations. SEAC employs a unique representation of the COMMZ to reflect the geography of the theater, and organizational support planning. Engineer requirements associated with facility and installation construction, repair, and maintenance are calculated based on the time-phased arrival of units, their movement and locations in the COMMZ, and the results of enemy units, their movement and locations in the COMMZ, and the results of enemy attacks as determined by an internal threat/damage sub-model. Engineer capability is based on the various skills found in engineer units used in the COMMZ, and distinguished by service and national affiliations. SEAC is a computer model based on object-oriented programming design techniques, and uses the process view of simulation to achieve its purpose. As a direct consequence of its hierarchical design, SEAC is highly modular and can be easily extended to examine related engineer problems in the COMMZ. This manual documents SEAC's code, data structure, and design. It also includes sections describing: various model output, --			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL DALE F. MEANS, COL, US Army Corps of Engineers, Commander/Director		22b. TELEPHONE (Include Area Code) (202) 355-2373/74	
		22c. OFFICE SYMBOL CEESC-XO	

UNCLASSIFIED

Item 19, Abstract (Continued)

considerations that should be used when assembling the various data bases, and the more prominent features of the software environment in which SEAC was developed.



Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

UNCLASSIFIED

DISTRIBUTION LIST

	<u>No. of Copies</u>
Commandant, US Army Engineer School, ATTN: ATSE-Z, Ft Leonard Wood, MO 65473-5000	1
Commandant, US Army Engineer School, ATTN: ATSE-DCD, Ft Leonard Wood, MO 65473-5000	2
Commander, 412th Engineer Command, P.O. Box 55, Vicksburg, MS 39180	2
Commander, 416th Engineer Command, 4454 West Cermak Road, Chicago, IL 60623	2
Commander-in-Chief, US Army Central Command, ATTN: CC J4/7-E MacDill AFB, Tampa, FL 33608-7001	1
Commander, Naval Facilities Engineering Command, 200 Stovall Street, Alexandria, VA 22332-2300	1
Commander, US Army Combined Arms Center, Ft Leavenworth, KS 66027-5000	1
Commander, US Army Engineer Division, Europe, APO NY 09757	1
Commander, US Army Corps of Engineers, Pacific Ocean Division, ATTN: PODEM, Ft Shafter, HI 96858	1
Commander, USAREUR, ATTN: DCSSENGR, APO New York 09403	1
Commander, US Army Logistics Center, Ft Lee, VA 23801-6000	1
Commander, US Army Southern Command, ATTN: SCEN, APO Miami 34003	1
Commander, US Army TRADOC Analysis Center, ATTN: ATOR-CAS, Ft Leavenworth, KS 66027-5200	2
Commander, US Army Training and Doctrine Command, ATTN: ATEN, Ft Monroe, VA 23651	1
Commander, US Army Western Command, ATTN: APEN, Ft Shafter, HI 96858-5100	1
Commander, US Forces in Korea, Assistant Chief of Staff, Engineer, APO San Francisco 96301-0092	2

	<u>No. of Copies</u>
Commander, US Forces in Korea/EUSA, APO San Francisco 96301	1
Commander, US Pacific Air Forces, Operational Analysis Office, Hickam AFB, HI 96853	1
Defense Logistics Agency, Defense Technical Information Center ATTN: DB-2D, Cameron Station, Alexandria, VA 22304-6145	2
Deputy Under Secretary of the Army for Operations and Research, Room 2E660, The Pentagon, Washington, DC 20301	1
Director, Joint Data System Support Center, The Pentagon, BF675, Washington, DC 20301-7010	2
Director, Office of the Secretary of Defense, Director, Program Analysis and Evaluation, General Purpose Programs, Land Forces, Rm 2B256, Pentagon, Washington, DC 20301	1
Director, Organization of the Joint Chiefs of Staff, J4 (Logistics), Deputy Director for Plans, Concepts, and Analysis, Washington, DC 20301	1
Director, Organization of the Joint Chiefs of Staff, J4 (Logistics), Washington, DC 20301	1
Director of Research and Development, US Army Corps of Engineers, Washington, DC 20314-1000	1
Director, US Army Concepts Analysis Agency, 8120 Woodmont Avenue, Bethesda, MD 20814-2797	2
Director, US Army TRADOC Analysis Center, White Sands Missile Range, White Sands Missile Range, NM 88002-5502	2
Headquarters, Department of the Army, Office of the Assistant Chief of Engineers, ATTN: DAEN-ZCM, Washington, DC 20310-2600	1
Headquarters, US Air Force, Assistant Chief of Staff, Studies and Analyses, Rm 1E388, The Pentagon, Washington, DC 20330-5420	1
Pentagon Library, Rm 1A518, The Pentagon, ATTN: Army Studies, Washington, DC 20310-6050	1
Readiness Directorate, Air Force Engineering and Services Center, ATTN: AFESC/DEO, Tyndall AFB, FL 32403	1

	<u>No. of Copies</u>
US Army Cold Regions Research and Engineering Laboratory, 72 Lyme Road, Hanover, NH 03755-1290	1
US Army Construction Engineering Research Laboratory, P.O. Box 4005, Champaign, IL 61820-1305	1
US Army Engineer Waterways Experiment Station, P.O. Box 631, Vicksburg, MS 39180-0631	1
US Army Engineer Studies Center, Casey Building 2594, Ft Belvoir, VA 22060-5583	13
Director, US Army Engineering and Housing Support Center, Bldg 358, Ft Belvoir, VA 22060-5516	<u>1</u>
TOTAL	57

ACKNOWLEDGMENTS

This manual was prepared to document the computer model developed as part of a sustainment engineering study conducted by the US Army Engineer Studies Center for the Republic of Korea/United States (ROK/US) Combined Forces Command (CFC). Mr. Robert Halayko conceived and implemented the model while serving as Project Manager for *Engineer Assessment Korea: Communications Zone Analysis*, working under the supervision of Mr. Lyle Suprise, Senior Project Director, and Mr. Dean Considine, Technical Director.

Acknowledgments are extended to those agencies and offices without whose invaluable information and assistance this model could not have been attempted. In particular, ESC wishes to thank: Headquarters, CFC, particularly the Assistant Chief of Staff, Engineer; Pacific Air Force, Operations Analysis; Joint Data Systems Support Center; US Army Facilities Engineer Support Center; and the 416th Engineer Command (Reserves).

This manual was prepared for publication by Mrs. Donna L. Norbutt, Ms. Stacia L. Hall, and Mrs. Collie J. Johnson. The editor was Ms. Cherie A. O'Neil. The graphics were prepared by Mr. John E. Hobbs and Ms. Linda G. Smith, under the supervision of Mr. Christopher Y. K. Lew.

CONTENTS

<u>Section</u>		<u>Page</u>
	DD FORM 1473	
	DISTRIBUTION	iii
	ACKNOWLEDGMENTS	vii
	CONTENTS	ix
	EXECUTIVE SUMMARY	xi
	LIST OF ABBREVIATIONS AND ACRONYMS	xiii
 I	INTRODUCTION	
	Purpose	1
	Scope	1
	Document Organization	1
	Background	2
 II	SYSTEM REQUIREMENTS	
	Hardware/Software Environment	4
	Data	4
 III	MODEL DESIGN	
	Model Overview	6
	Major Structural Components	6
	Major Processes	16
 IV	SYSTEM OPERATION	
	SEAC Execution	28
	Output	29
 V	MODEL FUTURE	
	Use of SEAC (a Digression)	31
	Future	31
 VI	POSTSCRIPT	
	Object-oriented Programming	33
 <u>Figure</u>		
1	Model Design Goals	3
2	SEAC Input/Output Structure	5
3	SEAC Methodology	7
4	Facility Classes and Subclasses	9
5	Installation Classes (Subclasses)	11
6	Region/Mapcell Example	12
7	Priority Groups	14
8	Air Threat Logic	18
9	SEAC Ranger/Commando Logic	20
10	Unit Directed Facility Requirements	21
11	Determining Facility Requirement	23

Figure

Page

12	Determining Damage Workload	25
13	Work Accomplishment	27
14	Execution Flow in SEAC	30
ANNEX A: SIMULA		A-1
ANNEX B: DETAILED SYSTEM DESCRIPTIONS		B-1
ANNEX C: DESCRIPTION OF SEAC'S DATA FILES		C-1
ANNEX D: USERS GUIDE TO SEAC		D-1
ANNEX E: SEAC REPORTS AND MESSAGES		E-1

EXECUTIVE SUMMARY

The Engineer Studies Center (ESC) has, since the late seventies, been analyzing existing OPLANs as part of an in-depth assessment of wartime engineering workload and the adequacy of engineer planning. Most of these studies concentrated on the forward combat zone (FCZ), and over time a well defined methodology evolved. Recent study sponsors, however, asked that ESC also look at engineer requirements in the communications zone (COMMZ). Since COMMZ engineer operations are far different from FCZ tasks, ESC needed a new analytic approach that realistically represented rear area engineer work, yet still was compatible with the FCZ analyses (to enable theater-wide assessments of engineer forces). The result is the Simulated Engineer Assessment for the COMMZ (SEAC), a powerful tool to conduct a large class of engineering analyses for COMMZ operations. SEAC employs a unique representation of the COMMZ to reflect the geography of the theater, and organizational support planning. Engineer requirements, associated with facility and installation construction, repair, and maintenance, are calculated based on the time-phased arrival of units, their movement and locations in the COMMZ, and the results of enemy attacks as determined by an internal threat/damage sub-model. Engineer capability is based on the various skills found in engineer units used in the COMMZ, and distinguished by service and national affiliations. SEAC is a computer model based on object-oriented programming design techniques, and uses the process view of simulation to achieve its purpose. As a direct consequence of its hierarchical design, SEAC is highly modular and can be easily extended to examine related engineer problems in the COMMZ.

This manual documents SEAC's code, data structure, and design. It also includes sections describing: various model output, considerations that should be used when assembling the various data bases, and the more prominent features of the software environment in which SEAC was developed.

LIST OF ABBREVIATIONS AND ACRONYMS

ADA.....air defense artillery
AFCS.....Army Facilities Component System
ALGOL.....algorithm
APOD.....aerial port of debarkation
ARLOC.....Army location

CATCODE... ..category code
CESP.....Civil Engineering Support Plan
CESPG.....Civil Engineering Support Plan Generator
COMMZ.....communications zone

EAK.....Engineer Assessment, Korea
ECAPB.....Engineer Unit Capability File
EHSC.....Engineering and Housing Support Center
ESC.....US Army Engineer Studies Center

FCZ.....forward combat zone
FEBA.....forward edge of the battle area

GEOLOC.....geolocation codes

HN.....host nation
HQDA.....Headquarters, Department of the Army

IFS.....Integrated Facility System

JCS.....Joint Chiefs of Staff

LOC.....lines of communication

MAINT.....maintenance
MEAB.....mean area of effectiveness for blast damage
MSG.....models, systems, and games
MSR.....main supply route

NB.....note bene (latin for "note well")

OOP.....object-oriented programming
OPLAN.....operation plan

PACAF/OA.....Pacific Air Force, Operations Analysis
PLNGFACT.....planning factor
POD.....port of debarkation
POL.....petroleum, oil, and lubricants

RAD.....Roentgen Absorbed Dose
RESTR.....restoration
ROK.....Republic of Korea
RPI.....real property inventory
RS&D.....reception, staging, and deployment

SEAC.....Simulated Engineer Assessment of the Communications Zone
Model
SOF.....strategic offensive forces
SPOD.....sea port of debarkation

TDA.....table of distribution and allowances
TM.....Technical Manual
TOE.....table of organization and equipment
TPFDD.....Time-Phased Force Deployment Data
TPFDL.....Time-Phased Force Deployment List
UTC.....unit type code

**SIMULATED ENGINEER ASSESSMENT OF THE
COMMUNICATIONS ZONE MODEL (SEAC)
(Documentation and Users Manual)**

I. INTRODUCTION

1. Purpose. This publication documents the computer system developed by the Engineer Studies Center (ESC) to determine engineer requirements and capabilities in the communications zone (COMMZ) during contingency operations.

2. Scope. The SEAC model is a simulation of engineer activities in the COMMZ. Its purpose is to evaluate the adequacy of engineer support of an operational plan (OPLAN). SEAC highly details engineer requirements and capabilities, which is also compatible with the engineer assessment methodology ESC has developed for its analysis of Forward Combat Zone (FCZ) engineer planning¹. SEAC uses state of the art computer programming techniques which greatly contribute to its design and successful implementation. The structure is based on a powerful modeling paradigm (object-oriented programming [OOP]), that has proven to be an elegant and productive means in which to construct models of complex systems. This document performs three major functions: model description -- to define the system; software documentation -- to list and describe the data and functions of the programs; and user's manual -- to assist users in exercising the model. The main paper introduces the model and describes its general capabilities, while the annexes contain the more detailed software, data base, and design specifications.

3. Document Organization. The document is organized as follows:

- * Main paper - contains a general description of the model and its major components. It also discusses the general requirements of the model and some of the reasons why particular features were included or designed.
- * Annex A - introduces several important features of SIMULA, the computer language used to implement SEAC.

¹US Army Engineer Assessment, Europe (United States Army Engineer Studies Center [USAESC], June 1981); *Analysis of VII Corps Combat Engineer Wartime Requirements* (USAESC, March 1983); *Analysis of V Corps Combat Engineer Wartime Requirements* (USAESC, December 1983); *Analysis of III Corps Combat Engineer Wartime Requirements* (USAESC, December 1984).

- * Annex B - contains listings, data definitions, and descriptions of SEAC's software components.
- * Annex C - defines the various data files used by SEAC.
- * Annex D - contains guidance and instruction to prepare input and scenario data for a SEAC application.
- * Annex E - provides listings and explanations of the many reports, messages, and warnings that have been built into the model to facilitate application development, execution, and interpretation of results.

4. Background. ESC has, since the late 1970's, been analyzing the adequacy of engineer support of various military operational plans (OPLANs). Initially, most of these studies concentrated on the FCZ and were confined to European studies. When ESC began doing such assessments outside of Europe, sponsors requested that sustainment engineering workloads also be included in the analysis. The decision to create SEAC was based on the exigencies of one of these studies and a perceived need for an in-house computerized system to analyze the engineer planning in the COMMZ. SEAC was begun to assist in assessing COMMZ engineer support in Korea.² Actually when the study began, ESC had no plan to use, much less construct, a model of rear area engineering. This changed once ESC saw the kinds and volume of data that it would have to acquire and manipulate. Consequently, the need for automated assistance became apparent. ESC originally planned to construct a dedicated model, i.e., one specifically designed to address Korea. However, it reconsidered the scope and application of the model as development proceeded, and as a direct consequence of the evolving design of the model. ESC decided that it could generalize the model to address a large class of COMMZ engineer problems. The objective was to have a flexible, in-house model that could be easily used or adapted. ESC often takes on studies that do not exactly fit other engineering models. In order to use SEAC on these prospective but likely analyses, ESC established the following revised design goals portrayed in Figure 1. Two different objectives were prescribed: one for the system to be simulated; and the other for the form of the implementation. The functional representation was to accurately simulate engineer operations and, among other things,

²Engineer Assessment, Korea: Communications Zone Analysis (USAESC, August 1987). Hereafter referred to as EAK.

MODEL DESIGN GOALS

1. Functional

- Permit flexible assignments of engineer assets.
- Retain detailed information about installations and facility assets and requirements.
- Use available data as much as possible.

2. Systemic

- Use a time-dependent, process-oriented modeling approach.
 - Employ good software engineering methods to facilitate the modification and extension of the prototype.
 - Separate code from data.
-

Figure 1

provide detailed reports on engineer requirements and capabilities at various levels of aggregation. ESC recognized, however, that changes to the model would arise with new study problems. The system, therefore, employs software engineering techniques to facilitate program development and modification. Because of its intended mutability, SEAC is sometimes referred to in this document as a prototype. This underscores its potential for adaptability. SEAC looks beyond the study for which it was developed, at special problems, which very likely could require extending or modifying the present model.

II. SYSTEM REQUIREMENTS

5. Hardware/Software Environment. SEAC was implemented on ESC's PRIME-750 minicomputer, which has 3 megabytes of main memory and over 300 megabytes of auxiliary disk storage. The model is programmed entirely in SIMULA, an ALGOL-based computer language.³ SIMULA is a general purpose, object-oriented programming (OOP) language, providing software features, such as inheritance, encapsulation, and strong typing. Discrete-event simulation models usually espouse one of three approaches to represent dynamic actions: event scheduling, activity scanning, or process interaction.⁴ SIMULA provides a simulation environment that supports the process view of model development and was, therefore, the approach adopted in SEAC. A SIMULA process can also contain all the relevant attributes and actions of the element, and thus provides an object-oriented environment (cf. SIMSCRIPT II.5).⁵ Choosing SIMULA was significant, because its power and features greatly facilitated development of SEAC. (See Annex A for an overview of SIMULA.) Implementing SIMULA requires rev18, or higher, of PRIMOS, which is PRIME's virtual memory operating system.

6. Data. As with any model, results are only as good as the input. SEAC uses many data files during a typical execution. Annex C describes the content, the source, and format of these data. Figure 2 shows the input/output structure of the model. From it one can see the number and variety of data that comprise the input portion of the system. Assembling the data prior to conducting actual analyses can vary. The amount of time is dependent, to a great degree, on outside organizations. In particular, the responsiveness of the sponsor in providing accurate threat, asset, and unit deployment data will determine how quickly the data can be orchestrated. One data source in particular is the Joint Chiefs of Staff (JCS), which has responsibility for

³Lamprecht, Gunther, *Introduction to Simula 67* (Friedr. Vieweg & Sohn, 1983, Braunschweig/Wiesbaden, 1983).

⁴Kreutzer, Wolfgang, *System Simulation Programming Styles and Languages* (Addison-Wesley Publishing Co., 1986) p. 41.

⁵Russell, E. C., *Simulating with Processes and Resources in SIMSCRIPT II.5* (CACI Inc., 1979).

SEAC INPUT / OUTPUT STRUCTURE

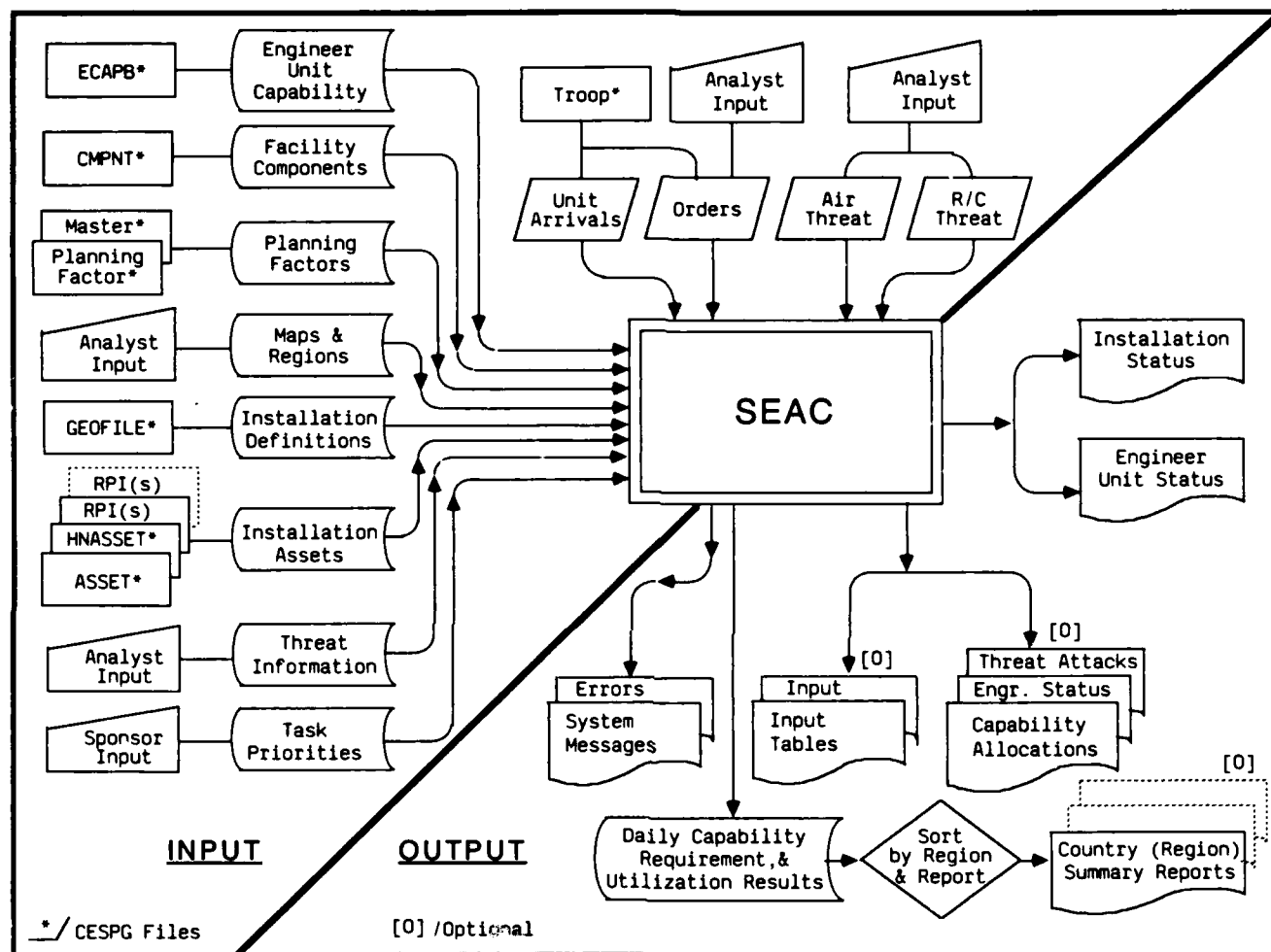


Figure 2

the Civil Engineering Support Plan Generator (CESPG) files that currently provide a major portion of SEAC's input.⁶

⁶Joint Operation Planning System (JOPS) Civil Engineering Support Plan Generator (CESPG) Users Manual, Computer System Manual [CSM UM 122-86] (Joint Data Systems Support Center, 1 April 1986).

III. MODEL DESIGN

7. Model Overview. SEAC determines COMMZ engineer workload and assesses how well engineer capability can meet the demands. Work is quantified by the size of the task, the manhours necessary to accomplish it, the facility class, type, and importance of the work being performed. Figure 3 portrays the major components of the model. Note in particular the central location of the facility block which is at the heart of SEAC. Most facility requirements are generated by units assigned to, arriving in, or passing through the theater. Planning factors determine which and how much facilities are needed.⁷ Factors are associated with supported population, equipment, particular units, and installations. Before new facilities are built, existing facility assets are checked to see if they satisfy requirements. If inadequate, construction may be scheduled, depending upon theater policies defined for the analysis. War damage repair tasks arise from either enemy ground or air attacks generated by SEAC's threat submodels. (Maintenance and restoration tasks, while not in the initial version of SEAC, would not be difficult to include once valid workload factors are developed.) The other side of SEAC deals with the capability of engineer units in the COMMZ to perform the work. Because SEAC identifies individual engineer units, this allows engineers to control their estimating capability to construct and repair facilities. Models and simulations can frequently be viewed as classes of data elements and the procedures that change these data from one state to another. The following sections first describe the major structural data elements within SEAC, followed by descriptions of the primary processes that manipulate the elements. Definitive descriptions of these, as well as the remaining system elements appear in Annex B.

8. Major Structural Components. SEAC is an example of object-oriented programming (OOP). The data components described below actually represent classes of model objects which, in turn, are examples of abstract data types.⁸ Objects are more than simply a particular piece of data somewhere in the host computer's memory (as someone familiar with FORTRAN, BASIC, or PASCAL might

⁷"Planning Factors for Military Construction in Contingency Operations (U)," Enclosure to JCS Memorandum MJCS 201-81 (13 October 1981).

⁸Liskov, Barbara, et. al., "Abstraction Mechanisms in CLU," Communications of the ACM, Volume 20 (August 1977), pp. 564-576.

perceive it). They are instances of specially defined data and operational templates that define the model using elements that approximate the real world system being simulated. Because of the ability to map real world elements to model objects, OOP's abstraction and design processes produce software that more closely resembles the target system. The following descriptions refer to terms of units, tasks, etc., that are analogous to their real world namesakes. To better understand the richness and power of object representations, the reader should examine the class definitions found in Annex B for the components below.

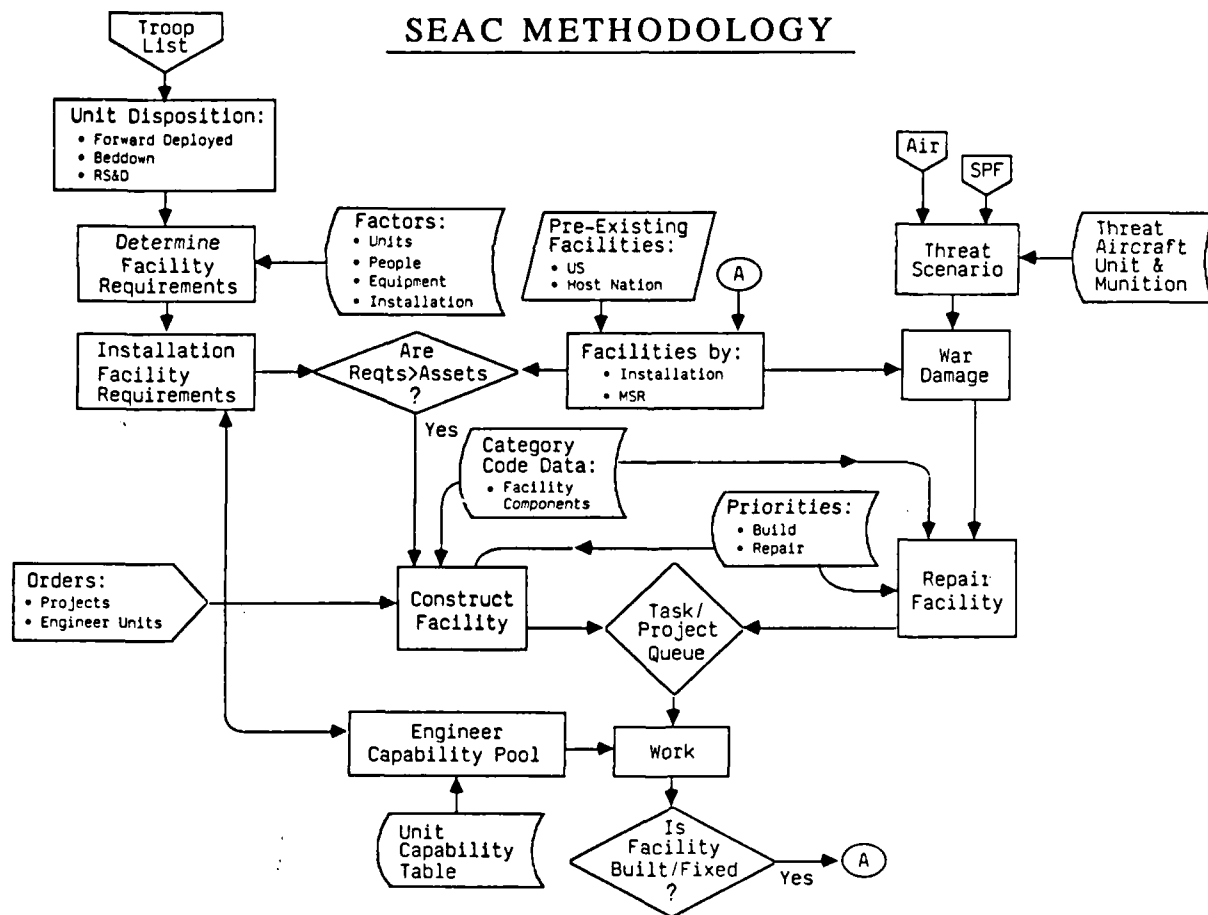


Figure 3

a. **Defining the COMMZ.** The stage on which SEAC acts is the communications zone of a contingency theater. The initial challenge in developing SEAC was to design a COMMZ representation that would support the overall engineer assessment process. ESC had previously developed a formulistic approach to analyzing combat engineer requirements in the FCZ. SEAC was compatible with that FCZ approach, while focusing on the sustainment engineering tasks that occur in the COMMZ. This was required despite the fact that the nature of engineer tasks in the forward and rear areas is quite different. SEAC had to be flexible enough to adapt to future studies in different theaters with possibly different analysis objectives. Moreover, the design had to balance the things necessary to evaluate engineer planning with limited data and practical limitations on how big the model could be.

(1) **Facilities.** Facilities play a central role in SEAC. Facility workload characteristics and data come directly from the several services' facility component systems⁹. In addition to identifying facility types, these data provide the daily work hours, by skill, that are needed to build or repair the facility; the minimum number of days to accomplish the task; and the unit of measure by which the facility is sized (e.g., 1,000 square feet or 60 kilowatts per hour). These data are available not only for constructing normal facilities, but also for emergency construction and for damage repair of existing facilities. Presently SEAC does not address restoration work, although such data is available. Comparable wartime maintenance data do not appear to be readily available. The versatility of SEAC's design, however, would allow ESC to assimilate restoration or maintenance data and procedures if needed, or when available. Facilities are identified by JCS category codes (CATCODES). Within SEAC, similar facilities are grouped into general classes for buildings, surfaces, transportation networks, utilities, petroleum, oil, and lubricants (POL). Surface facilities, for example, include runways, hardstands, taxiways, parking areas, etc. Figure 4 gives the mapping of category codes into SEAC classes and subclasses. This hierarchy of facilities enables SEAC to associate different operations with particular facilities. The primary reasons for this structure are war damage estimation

⁹Army Facility Component System - Design Manual for Architect-Engineers Guidance, HNDM-1110-1-1 (US Army Corps of Engineers, Huntsville Division, September 1980).

FACILITY CLASSES AND SUBCLASSES

Class	Subclass	JCS CATCODES
Surface	Runway	111
	Pavement	112, 113, 116, 153
	Hardstand	852
Petroleum	Pipeline	125
	Tanks	411, 124
	POL facilities	121, 122, 123, 126
Building	Communications	131, 133
	Operations	149
	Shops	211, 213-219
	Storage	421, 425, 431, 432, 441, 442
	Medical	510, 540, 550, 560
	Administrative	610
	Quarters	721, 722, 724, 725, 730
	Revetments	149
Utility	Piers	151, 152, 159, 163
	Water	841, 842
	Waste	831, 832
	Power	811, 812
Transportation	Roads	851
	Railroads	860

Figure 4

and repair task estimation. Facilities have different criteria for damage; some need only be within a weapon's blast radius, while others will be damaged if within the fragmentation area.¹⁰ To begin with, SEAC can address these differences because it contains a detailed threat submodel that simulates individual enemy attacks against facilities (see 9.a.). The effect of dropping a general purpose bomb is not pre-determined; it will depend upon the class, size, and vulnerability of the facility and the characteristics and impact location of the bomb. Even how damage is characterized can be different: runway damage is measured in craters, a building is by square

¹⁰Communist-World Weapons Effectiveness, Selection, and Requirements Handbook (U), DDB-2660-21-80, Change No. 1 (Defense Intelligence Agency [DIA], February 1980).

feet, but a pipeline may be in linear feet. Repair procedures can also be characterized differently.

(2) Installations. An installation is "...a group of facilities in the same vicinity that supports particular functions..."¹¹ Installations are another SEAC COMMZ building block. Other than lines of communication (LOC) facilities (i.e., roads, railroads, and pipelines), all facilities are located at the installations. Units are also located at installations. We shall discuss unit reception, staging, and deployment in more depth in a later section. It is important, however, to note that the only unit positional information available to SEAC are the port of debarkation (POD) and destination geolocation codes (GEOLOC) derived from the time-phased force deployment data (TPFDD). COMMZ units are those that are either forward deployed, or are assigned to installations in the COMMZ. SEAC distinguishes between COMMZ units and FCZ-bound units, although both types may at any time be located at an installation. To do this, SEAC has categorized installations based on their functions. An installation designated as a staging base distinguishes between those units that are assumed to become permanent party, and those units that will deploy to the front. Installation classes are defined as well for air bases, ports, etc. (see Figure 5). These classes enable SEAC to tailor operations at a particular site to better emulate what would actually be happening. SEAC defines requirements in terms of the facilities that the units, their personnel, and their equipment need according to applicable planning factors (see 8.b.). Facility requirements can also be ascribed to an installation class. For example, no air force unit has a factor that requires a runway; a runway is presumed to exist along with certain other facilities that comprise a baseline air base. Sometimes, however, the facility asset file may be incomplete, and there is no runway defined. Also, the COMMZ or theater can expand creating a requirement to construct an installation from scratch. Installation planning factors can be used in these situations to establish or verify minimum facility requirements. Those requirements can then be translated into construction jobs necessary to satisfy needs. Lastly, installations contain lists of engineer tasks, and

¹¹AFCS Design Manual (US Army Huntsville Engineer Division, HNDEM-1110-1-4, 1 September 1980), p. 1-3.

INSTALLATION CLASSES (SUBCLASSES)

Air Bases	(APODs, collocated)
Ports	(SPODs)
ADA sites	
Camps	(RS&D centers)
Electronic sites	
Depots	
Other	

Figure 5

engineer units available to work on those tasks (see the discussion of Orders and Work).

(3) Region/mapcell. A major innovation in SEAC was the design of an abstract, hierarchical representation of the theater. This design provides an extremely compact, and accessible framework for the COMMZ, which captures those features necessary to represent engineer operations for any theater configuration. It does so using two different elements -- regions and mapcells. Mapcells define a unique partition of the theater. One could overlay the theater with a square grid, and each grid cell could then be defined as a mapcell. Mapcells, however, needn't be regularly shaped, nor even the same size. Their primary purpose is to fix the location of installations and main supply routes (MSRs). A mapcell can contain lists of installations, MSR facilities, and MSR tasks (installations contain all other tasks). Complementing mapcells in SEAC's theater representation is the region, a special SEAC object that groups mapcells or other regions (in which case they are called subregions). Regions enable SEAC to emulate the general area support role of engineer troops, particularly engineer combat heavy battalions. A region can contain subregions, mapcells, and engineer units. Such units are available for any work that arises in any of the mapcells that belong to the region or, derivatively, in any of its subregions. A simple region/mapcell theater definition is portrayed in Figure 6. It is a multi-level structure. The bottom level is the plane that contains the mapcell definition of the

COMMZ HIERARCHY IN SEAC

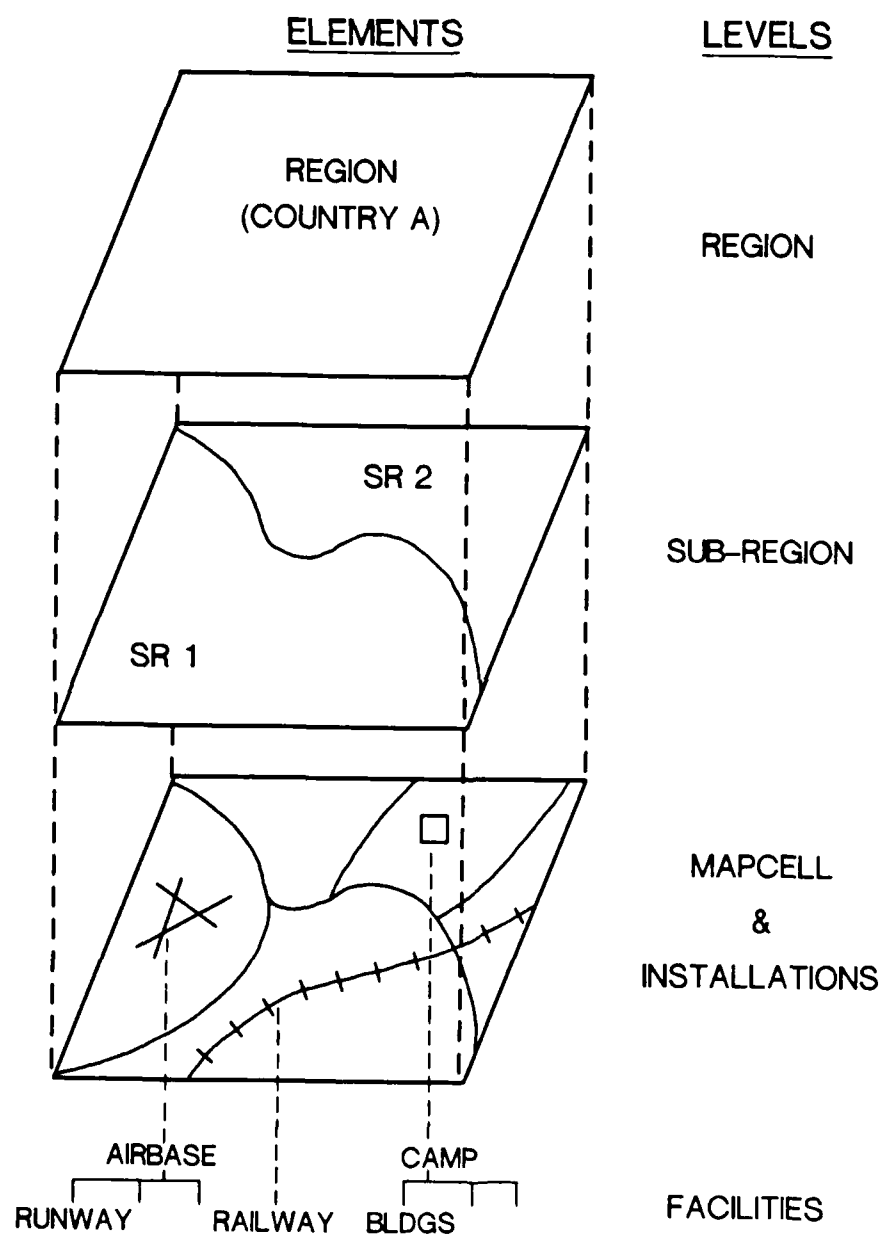


Figure 6

theater, and the upper levels or planes are the region and subregion definitions.¹²

b. **Planning factors.** SEAC evaluates engineer planning. The first step in this process determines the facilities needed by friendly forces and whether existing facility assets satisfy those needs. (We exclude damage, since that essentially modifies assets.) Planning factors determine what various elements need or use. An air force squadron, for example, requires a certain amount of parking space for each plane; housing and troop support facilities for its personnel; POL storage for fuel; etc. What types of facilities and in what quantities are dictated by planning factors. In our example, there could be several factors at work: one that translates unit populations into troop support needs; one that indicates how many, and which types of planes the unit has; and one (assuming there is one type of plane) that indicates what each plane will require. Some factors may be institutionalized, such as the amount of parking space for an F-4, or they may be plan- or theater-dependent, such as the amount of water consumed per person. Factors prescribing facility requirements can be defined for units, people, equipment, and installations. Factors can also be universal, or can be differentiated by country and by service. (See discussion of Factor file in Annex C, Data Files.)

c. **Priority/policy.** Part of the impetus to develop SEAC was the need to design a COMMZ model that was compatible with the risk assessment methodology ESC had developed for the analysis of FCZ engineer planning. Assigning priorities to each and every engineer task is at the heart of this process. Every construction and repair task in SEAC has a priority associated with it. The process employs four priorities: vital, critical, essential, and necessary (see Figure 7 for associated characteristics). A priority relates the importance of the task to the overall success of the OPLAN. A

¹²The regional hierarchy is analogous to the political boundaries of any country. Suppose we defined mapcells as every county in the lower forty-eight. We could then define a regionalization as follows: at the top would be the US; it contains subregions of New England, Middle Atlantic, Midwest, Pacific Northwest, etc.; and each of those contains subregions corresponding to the states that are traditionally linked to the areas (subregion New England contains subregions: Maine, New Hampshire, Vermont, etc.). And finally, each state subregion would have a list of every county that is within its boundary.

PRIORITY GROUPS

Short Title	Implications of Nonsupport
Vital	Jeopardizes the existence of friendly force High loss of life Early defeat
Critical	Failure of friendly force operations Increased probability of defeat
Essential	Short-term degradation in sustainability Significant equipment/materiel losses
Necessary	Long-term degradation in sustainability Moderate equipment/materiel losses

Figure 7

vital task is one that must be done if the OPLAN is to succeed. A necessary task is one that should ultimately be done, but whose impact, if not done, does not threaten a successful campaign. The results of the assessment provide an estimate of engineer capability to support the OPLAN under study. In the initial version of SEAC, new construction priorities were based on the installation class at which the new facility was required. This was, however, largely due to the particular needs of the study within which SEAC was developed.¹³ Repair task priorities are associated directly with the class of the facility that was damaged. In addition to designating the priority, the country can also be designated for applications with multinational play. Directly associated with priorities is the "policy." A policy defines,

¹³Broad assignment of priorities by installation class results in equating construction of mess halls with operational facilities. This probably does not reflect what would actually happen. The better solution is to assign construction priorities based on facility class. The impact on the model is negligible.

either implicitly or explicitly, what and how much work will be generated for a task. An implicit policy is enacted by simply not including a priority entry for a country's specific construction or repair task. If, for example, there were no entry for US railroad repair, SEAC would not create a task, even if railroad damage occurred. The presumption is that either such damage would be ignored or that someone else would do it. An explicit policy is denoted by giving the percent of a task that a country is expected to accomplish.¹⁴

d. **Engineer units.** SEAC allocates engineer assets for workload in the COMMZ. The source of engineer resources is the various military and civilian engineer units that are planned for the OPLAN. Engineer units are individually identified by unit type code (UTC), and have attributes for personnel strength, time in theater, country, and service. The UTC is used not only as an ID but also as an entry to an engineer unit capability table, which establishes the normal daily capability, defined by skill. Capability can be modified in several ways. One is a general availability factor defined for each country that can be used to represent sickness and casualty losses. The other is a phase-in factor that allows for a gradual (10-day) and reasonable increase in unit productivity. In addition, if a unit's strength is different from the reference strength of the capability table entry, then the unit's nominal capability is proportionately modified. Units are normally inserted into the theater after being read in from the Troop file. Engineer units are, however, an exception because they are controlled by entries on the Orders file. Orders enable a user to insert, move, and remove engineer units from the simulation. Units can be assigned to either installations or region/subregions. A unit can work on any task within the conceptual boundary of its location. If assigned to an installation, it can only work on tasks at that installation no matter what the workload is at nearby installations.¹⁵ By explicitly identifying each engineer unit, maximum flexibility and control are

¹⁴This type of policy enables a user to directly use the host nation support percentages as found in the Army Force Planning Data and Assumptions, FY 1988-1997 (DA, August 1987).

¹⁵This deals rather nicely with interservice support. By assigning Air Force, Marine, or Navy engineer units to installations, we can assure that their responsibilities will be focused. Army units, such as port construction companies can also be assigned to a port to localize its utilization, while combat heavy battalions (or host nation units?) can be assigned to a region and, thus be able to support any requirements.

maintained. General and direct support concepts can be emulated. Moreover, the door is left open for the possibility of expanding certain specialized tasks requiring specific unit types and support.

e. **Tasks.** SEAC requirements and capability come together in model objects called tasks. SEAC combines construction or repair requirements with information from the various functional component systems to create a work task. Tasks are added to lists of tasks maintained at the installation or mapcell where the facility is or will be located and, consequently, where the task arose. The task remains in the job queue until enough engineer capability can be applied to complete it. Tasks have the following attributes: average daily skill requirements to complete, minimum number of days to complete, priority, and a reference to the facility (a separately maintained SEAC object) that spawned the task. Because some tasks must either take several days to complete or, because of insufficient capability, be accomplished over time, there is also an internal parameter that indicates the progress that has been made on the task. When a task is completed, it is removed from the list and destroyed, but not prior to making adjustments in the effected facility (either removing a damaged flag, or indicating that construction was completed).

9. **Major Processes.** While elements define what is manipulated in SEAC, processes define how those elements are created and modified. The major processes deal with determining facility requirements, carrying out attacks by enemy forces against installations and facilities, and applying engineering capability to the workload.

a. **War damage.** Determining facility requirements, although it involves a considerable amount of supporting data, is rather straightforward in operation. War damage repair, another responsibility of engineers, is not as easy to calculate. Facility requirements proceed in lockstep with the movement of units into and through the COMMZ. The Troop file (TPFDD) supplies the dates and locations to track those movements. There is no such script yielding a detailed list of facilities, locations, and times for war damage. Unfortunately, we do not know the enemy's plan nor do we know how successful it will be and how success or failure will influence subsequent attacks. Because of this uncertainty, other engineering models distributed damage over

time, and over installations.¹⁶ It seems, however, that while the magnitude of damage might turn out to be reasonable in the CESPG, the theory that the enemy would attack all installations (and for only the first 30 days of the war) seems rather tenuous. Because of the resolution and flexibility of SEAC, ESC availed itself of intelligence estimates of enemy capability and intentions to define when and where damage would arise.¹⁷ In both the air and ground threat processes described below, enemy attacks are predesignated by time, size, composition of attack, and target destination. SEAC calculates the success of these attacks and the damage that results.

(1) Air War. SEAC models enemy air operations in quite some detail. Each plane is represented within the model, as is each bomb that the plane carries. The logic used to generate sorties was derived from a report prepared for ESC by the Pacific Air Force, Operations Analysis (PACAF/OA).¹⁸ While OA's estimate of sorties against air bases was necessary for ESC's assessment, it did not go far enough. ESC had to translate those sorties into facility damage, and it had to address all classes of COMMZ installation targets (see annexes A & B), not just air bases. The resulting adaptation is shown in Figure 8. First, SEAC defines the type of planes and the ordnance, then consults a list of air missions to see when the next attack is scheduled (see Annex C). The model conducts an attack against designated installation targets, taking plane availability and sortie success probabilities into account. Upon successfully reaching its target, each plane picks a facility target (based on attack apportionment directives), and if it is successful in hitting the facility, the amount of damage is recorded for that particular

¹⁶See CESPG, pp. 2-63.

¹⁷A fully automated system would be desirable. This system might allocate sorties, simulate the air war, and readjust airfield asset dispositions. After considering several alternatives, ESC decided to employ a hybrid system rather than to attempt an automated system. Reasonable threat scenarios, corroborated by intelligence sources, would be posited. The success or failure of these planned missions would be evaluated using the methodology used by PACAF/OA.

¹⁸ESC received a revised air threat scenario from PACAF's Operations Analysis Office (OA) as it began considering the need to develop an automated system for EAK-COMMZ. OA provided not only an air scenario acceptable to ESC's sponsor but also an outline of how the sortie rates were estimated and guidelines on targeting and hitting air base facilities (Air Base Damage Repair Requirements in a Korean Conflict (U) (Headquarters, Pacific Air Forces, May 1985).

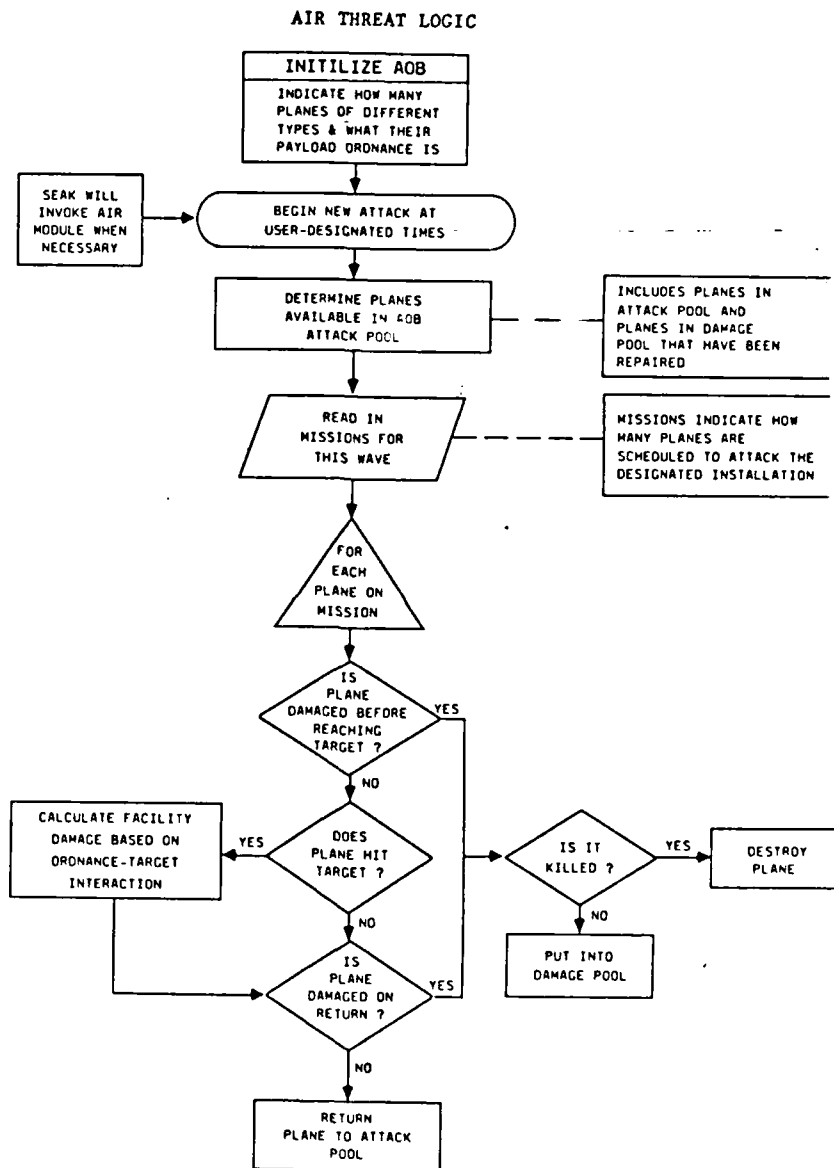


Figure 8

facility. The probabilistic events that determine the success or failure of missions represent the stochastic portions of the SEAC Air module.

(2) Ranger/Commando. One of the major concerns in the study for which SEAC was created, was facility damage resulting from enemy rear area attacks, particularly by ranger or commando units. The effect of these forces in the rear, and more importantly their effect on engineer workload, i.e., repairing facility damage, was ill-defined but of concern for that very reason. Having already developed the air logic, ESC used a similar scheme for ranger/commando type operations. Figure 9 outlines this logic. The major difference between the two threat modules is that while planes can be used over and over, the ranger/commando teams are considered to be expendable. After a team reaches its destination and attacks its target facility with whatever explosive materiel it carried, SEAC deletes the team. This is not to say that the team ceases to be a problem in the COMMZ. Once the team uses its defined load, however, there is no realistic way to estimate what additional facility damage it may or may not cause in later days. Another point worth mentioning is that a team may avoid being destroyed during insertion, but it may not necessarily land on its target installation due to navigational or equipment error. SEAC places that team in the rear and assumes that it will then attack either installations or MSR targets if it avoids being eliminated by rear area security forces. Rear area security forces will, however, be looking for those teams, and many teams will be destroyed before carrying out any attacks. As was the case for the Air module, the strategic offensive forces (SOF) module employs various user-defined probabilities to generate event outcomes.

b. Reception, staging and deployment (RS&D). While creating SEAC, ESC was concerned with estimating RS&D requirements. RS&D is the process by which deploying personnel enter a theater (at a port of debarkation [POD]), are temporarily assigned to a reception camp, move to a staging base, marry up with any equipment, and then move to their assignments (in the case of most combat units, that would likely be to the FCZ). CESPG's inability to adequately treat this area was another reason why ESC constructed its own model. Figure 10 describes how the RS&D and all types of units found on the Troop file are processed. The process begins by reading the units identified by UTC entering the theater that day. An important assumption is that if a

RANGER/COMMANDO LOGIC

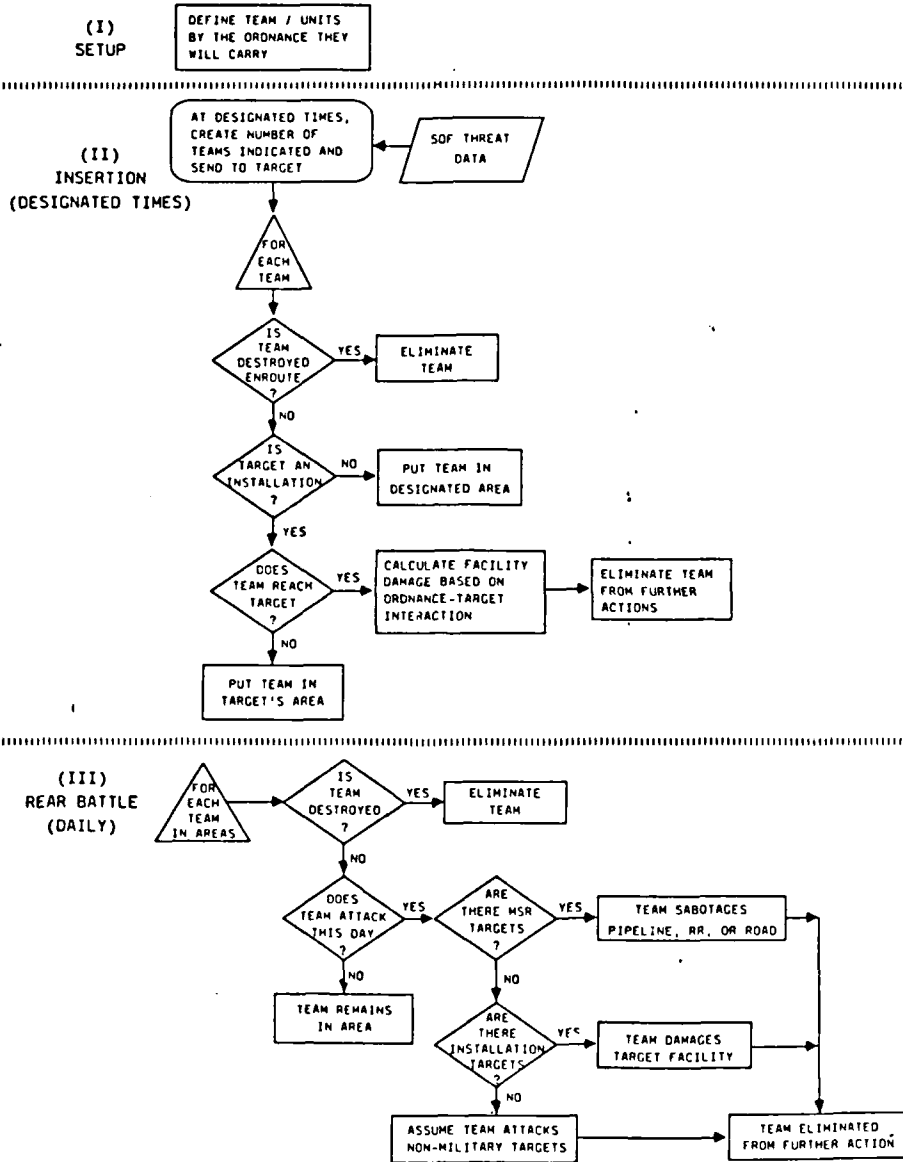
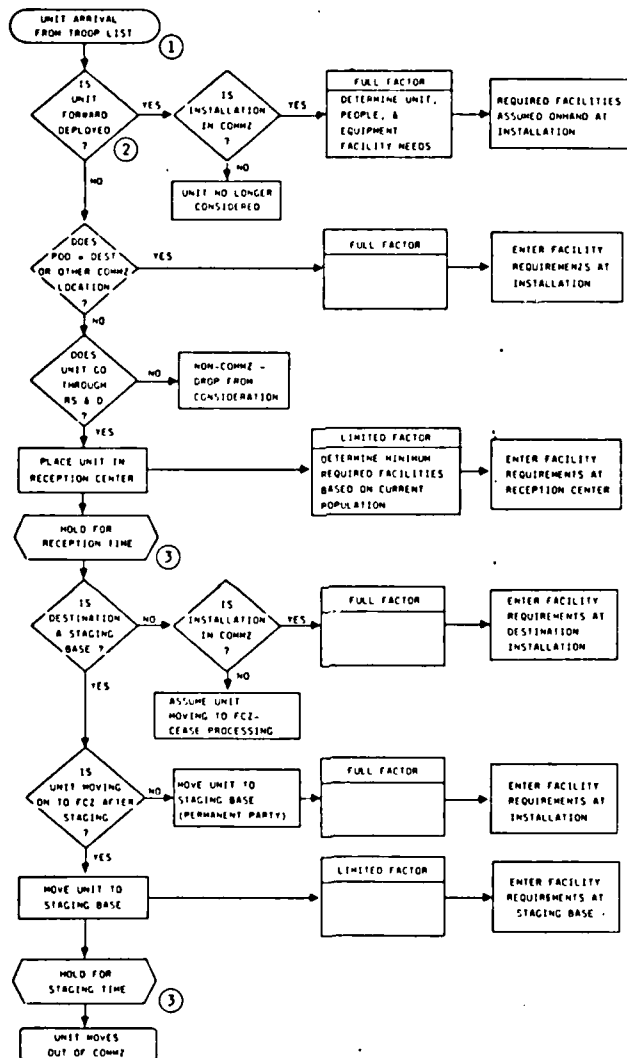


Figure 9

UNIT DIRECTED FACILITY REQUIREMENTS (Reception, Staging, and Deployment)



- ① TROOP-UNIT RECORDS CONTAIN ARRIVAL TIMES, PORT OF DEBARMENT GEOLOC (POO), AND DESTINATION GEOLOC (DEST).
- ② FORWARD-DEPLOYED UNITS ARE INDICATED BY ARRIVAL TIMES GO-DAY.
- ③ SPONSOR-DEFINED (FOR EAM, RECEPTION = 2 DAYS AND STAGING = 7 DAYS).

Figure 10

unit is already in the COMMZ (having an arrival date less than 0) when the scenario begins, SEAC determines the needed facilities and assumes that such facilities exist. If the unit deploys to the theater, it first goes to a reception center (designated for units of a particular service arriving at a POD) for a defined time. If a deploying unit's port of debarkation is the same as its destination, then the unit is immediately assigned to the installation and a unit-level factor check is made (where the unit, its equipment, and its people could generate facility needs). The unit at the reception center generates a need for reduced (limited) troop support based only on the strength of the unit. The needed facilities may have already been built for units which have since departed. When the unit moves to its destination, one of three courses is possible: if the destination geographical location (GEOLOC) is outside the COMMZ, then the unit receives no further consideration; if the unit goes to a COMMZ destination, then it is assumed to be assigned to that installation, and a unit-level factor check is made; if the unit's destination is identified as an RS&D center (staging base) and the unit is further identified as likely moving to the FCZ (as would be the case for Corps units), then only a reduced need for facilities is generated (as was done for the reception centers) and after another defined period of time, in which the unit is in the staging base, it is assumed to move on to the FCZ and is no longer considered by SEAC.

c. **Task generation.** As stated in the overview and implied in the previous process descriptions, the objective of these rather complicated manipulations is the systematic generation of engineer work -- either new construction or repair.

(1) **New construction.** It has been shown how facility requirements arise from processing the Troop list units. Figure 11 shows that in addition to unit directed facilities, there are two other ways to indicate needed facilities. One way is to use the method described in the planning factor section in which a particular installation type is assumed to have certain facilities by virtue of its class. An air base has a runway, some operations buildings, some fuel storage, etc.; otherwise, it would not be an air base. SEAC defined air bases, air defense artillery (ADA) sites, and electronic sites as having certain minimum facilities. (The initial impetus for this facility requirement device was the absence of facility data for

DETERMINING FACILITY REQUIREMENTS

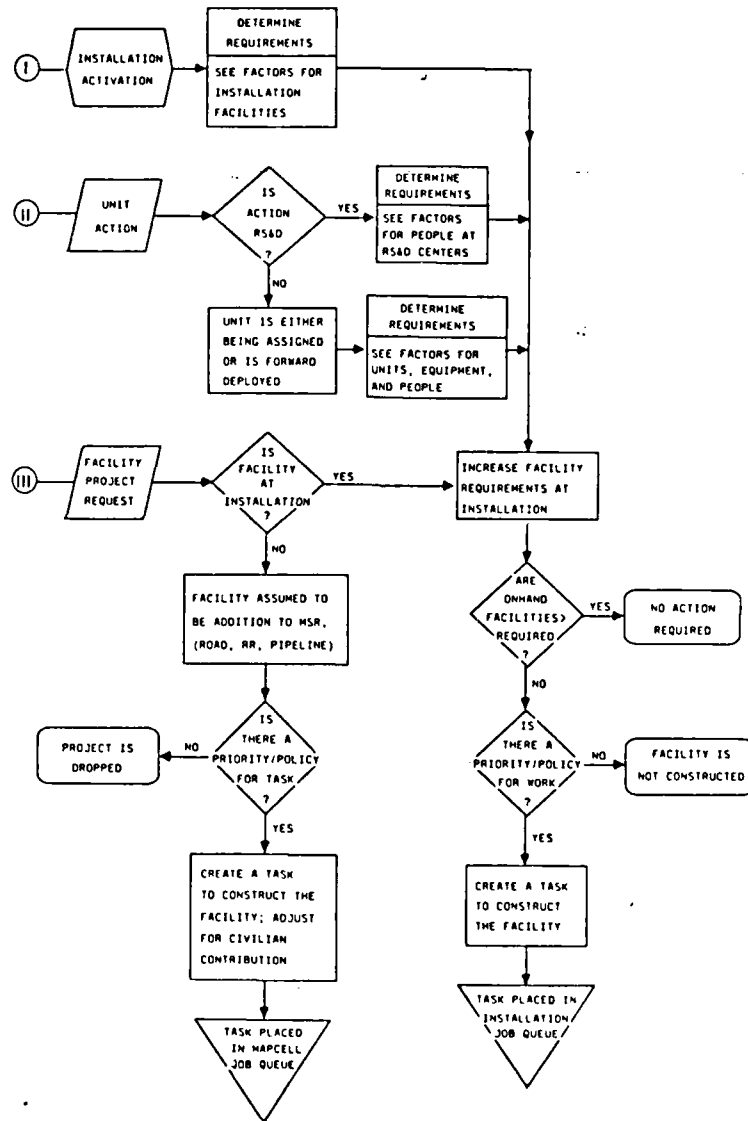


Figure 11

Republic of Korea (ROK) installations, other than runway dimensions and POL storage capacities. It was necessary to prescribe facilities in order for threat attacks to generate damage.) The other means to enter a facility requirement is through the explicit insertion of it in the Orders file. The Factor file might not generate a requirement for exceptional facilities (e.g., a tactical marine terminal or a contingency pipeline). The project order (as opposed to the engineer unit order) increases the requirement for a facility if it is to be built at an installation, or actually generates a construction project if the project is for an MSR facility. The last thing that is checked before a construction task is created is whether there is a construction priority/policy for the installation. If there is, then the task is scheduled with the designated priority and adjusted for any policy-implied civilian assistance.

(2) Damage repair. Determining damage repair is basically a two-phase process. ESC has already described the first stage, namely, attacking and damaging facilities by threat forces. During these attacks, SEAC tracks the damage for each targeted facility. Damage can be defined differently for different facilities. Damage on a runway, taxiway, or hardstand is defined in terms of craters to be repaired; a pipeline may have a length of pipe to be replaced, while buildings (the most common facility) will accumulate the amount of square feet that has been damaged. When the model looks at damage to generate repair tasks, it first looks to see if the facility has been destroyed. SEAC can play damage thresholds for each facility subclass to indicate when damage is to be considered superficial, reparable, or structural. If the model considers the amount of damage structural, the facility repair is regarded as a total loss. If the model assumes that the facility is destroyed, then it is not repaired; rather, the on-hand amount of that facility at that installation is reduced by the size of the destroyed facility. (NB: it will be reconstructed only if there is a requirement for that facility.) If repair is indicated, then the damage repair priority/policy of that facility subclass is checked, and if there is a policy to repair, then the task is scheduled. As in the case of new construction, adjustments can also be made for any implied civilian assistance. The process steps are shown in Figure 12.

DETERMINING DAMAGE WORKLOAD

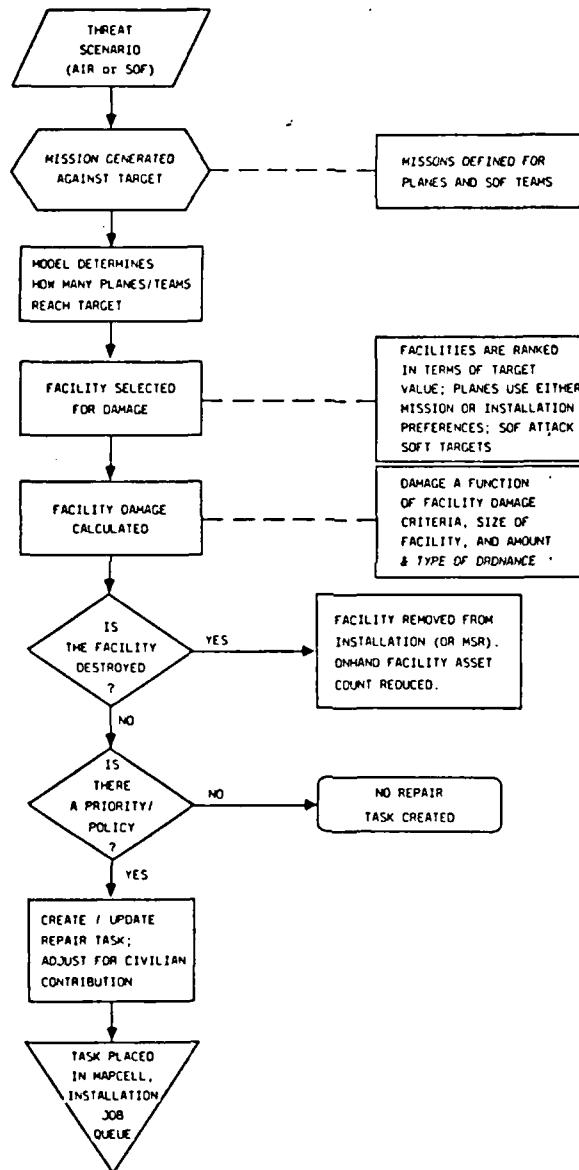
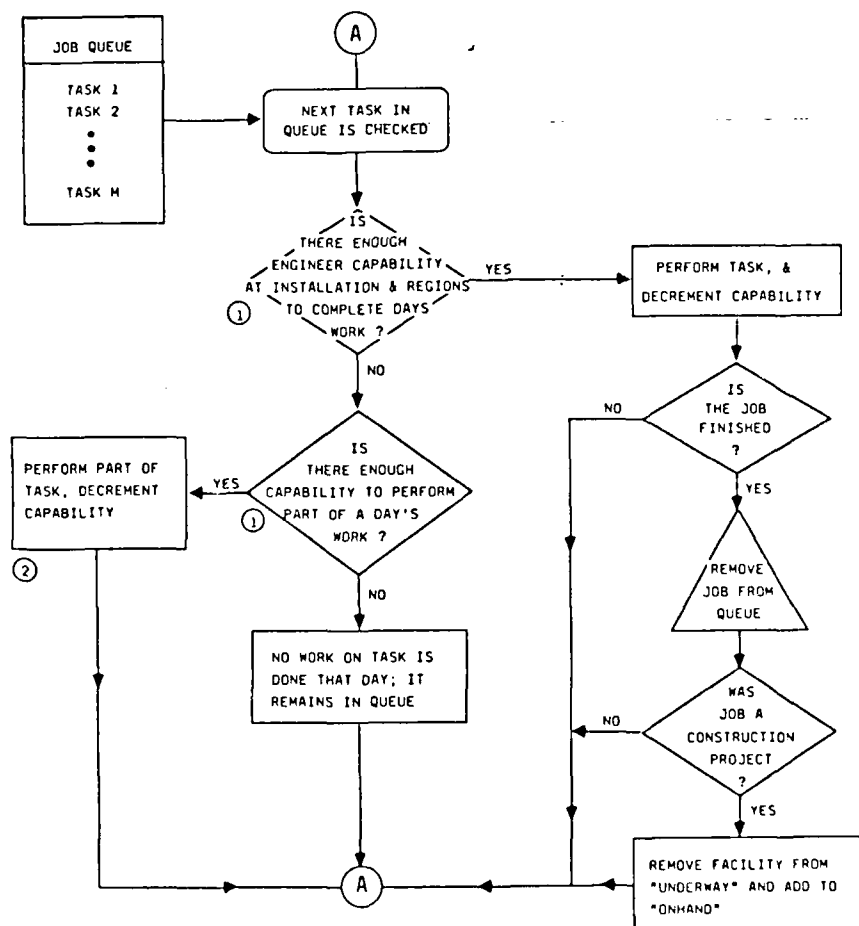


Figure 12

(3) Maintenance. Facility maintenance during wartime seems not to have received a great deal of study, especially in contingency operations. Operational facilities will necessarily have to be kept functioning. ESC designed SEAC to easily integrate maintenance into the model's task structure, but data to support such estimates were unavailable or have eluded the efforts of ESC to find. ESC made a compromise in the Korea study where facility engineers were presumed to be dedicated to maintenance, and were not made available for construction or damage repair work. Although this satisfied the needs of that study, it must be viewed as a temporary and unsatisfying accommodation.

d. **Work accomplishment.** A comparative analysis of how much of the model is devoted to generating workload versus how much of the model is spent keeping track of engineer capability and applying it to offset workload would probably show that 80 percent goes for the former and only 20 percent for the latter. Figure 13 shows the straightforward way in which capability is applied to workload tasks. The work process makes four separate passes through the COMMZ when scheduling work. The first pass attempts to perform all vital tasks, the second all critical, the third all essential, and the fourth all necessary. When SEAC selects a task from the job queue, it first checks to determine if engineer capability at the installation can do the task. If there is sufficient capability at the installation, the work is considered performed, engineer capability is reduced by the amounts necessary to perform the work, and progress is indicated for the task. If assistance is needed, then the engineer support at the region to which the installation belongs is queried. If the capability is still deficient, then successive superior regions are queried until either the task (or a portion thereof) can be performed, or no capability is expended on the task that day and it remains in the job queue unchanged. If work is performed, the task checks to see if it has been completed (remember that some projects may be several days in length). If the finished task was a new construction job, then the particular on-hand facility assets at the installation are increased by the size of the completed facility. SEAC recomputes the capability of installation and region engineers on a daily basis. SEAC explicitly tracks all engineer units (otherwise it would not be able to add, delete, or move engineers through

WORK ACCOMPLISHMENT



① SKILL SUBSTITUTION IS ALLOWED (HORIZONTAL & VERTICAL HOURS CAN BE USED TO SATISFY A DEFICIENCY OF OTHER HOURS).

② PARTIAL PERFORMANCE ASSUMES THAT A UNIFORM PORTION OF THE DAILY REQUIREMENT IS DONE.

Figure 13

orders), and determines their individual strengths before pooling their capabilities at different locations.

IV. SYSTEM OPERATION

10. SEAC Execution. SEAC is relatively self-contained, i.e., SEAC is a single program that processes all data and generates all output. Older computer models frequently divided systems into various programs with specialized purposes. Data files were the inter-program means of communication. Such an approach was not always preferred, but was the only way to proceed because of memory and storage constraints, or limitations of the software environment. This was not a problem in developing SEAC. The virtual memory system in which it was developed, coupled with the dynamic memory allocation of SIMULA, relieved the need to do things piecemeal. Despite larger memory availability, one can exceed capacity. The model would almost certainly have to be run on portions of the theater and troop list if it were used to assess the European theater, which is considerably larger than Korea. The following paragraphs discuss the various preparatory actions that precede executing SEAC and the sequencing of processes within the model itself.

a. Preparation. As shown in Figure 2, SEAC accepts a considerable amount of data. Unlike the very restrictive data file update and creation procedures of the CESP (harkening back to batch processing days), SEAC data files are expected to be manipulated using a normal system editor. ESC benefited greatly by having a powerful EMACS full-screen editor.¹⁹ EMACS not only provides an extensive list of editing commands, but also a programming capability (the now ubiquitous macros) that reformats, extracts, and manipulates, and saves countless hours as well as obviating the need to write programs to perform the operations. The ability to use only an editor to prepare the data depends, however, on the form of the available source data. A case in point is the asset data. SEAC uses JCS CATCODES for facility identification; the Army's Integrated Facility System (IFS) data base shows facility assets by AFCS codes. Though similar, the two systems are not the same. A program was written to convert the IFS data to the other form. Such situations arise, but are often dictated by the particular requirements of the

¹⁹EMACS Reference Guide, Revision 18.3, (PRIME Computer, Framingham Massachusetts, 1982).

study and the condition of available data. For a description of the form, contents, and sources of the various input see Annex C.

b. **Program sequencing.** Because it simulates activities in the COMMZ, SEAC is tied to the timetable of the OPLAN. Conceptually, one can think of SEAC as having four phases: initialization, activation, daily processing, and termination. Initialization sets up the theater and the various reference tables used in the simulation. The phases and their components are shown in Figure 14. Activation and daily processing run concurrently. The former describes the invocation of the various event lists for engineer unit and project declarations, unit arrivals, and threat attacks, while the latter describes the sequence of events that occur each day and are part of SEAC's normal operations. The events make changes in the COMMZ that result in various responses. Termination refers to those operations that occur upon completion of the scenario and are largely the production of various summary reports.

11. **Output.** SEAC tracks many items during the course of execution. There are many reports and error and status messages produced by the system. The output is file-based but can be directed to any device -- printer, terminal, or disk. The user can direct where errors, reports, and execution messages (log entries) are to be written, or accept default assignments made by SEAC. As with the rest of SEAC, the current list of reports is not intended to be definitive. One of SEAC's goals was, in fact, to produce detailed status reports at particular locations to enable US and allied forces to negotiate facility needs based on foreseeable requirements. The model maintains the data to be able to do that; it just has not been done. In addition, if any of the possible enhancements listed below are added to the model, there would likely be new reports written to accompany their operations. The reader can refer to Annex E which presents examples of all the current output.

EXECUTION FLOW IN SEAC

Phase	Activity/Process
Initialization	Construct Data Tables .engineer unit capabilities .facility components (CATCODEs) .planning factors Define Theater (regions & mapcells) Create and Locate Installations (GEOLOCs) Create Facilities and Place into Installations Define Threat (characteristics & quantities) Enter Policy and Priority Assignments Initialize Various Runtime Control Parameters
Activation	Initiate Event Streams .Troop list processing .Air mission generator .SOF insertions .ORDERS (engineer assignments & projects)
Daily	Read Orders Process Arriving Units From Troop List War Damage* Evaluate Facility Requirements Generate New Engineer Tasks .Damage repair .New construction Determine Available Engineer Capability Work (new & backlogged tasks by priority) Regional Result Retention War Damage*
Termination	Prepare Reports .Installation status .Engineer unit status .Summary COMMZ reports

*Attacks can occur at any time during the day.

Figure 14

V. MODEL FUTURE

12. Use of SEAC (a Digression). SEAC was designed to be a tool to analyze COMMZ engineer planning. In its detailed representation of the COMMZ, estimation of requirements, and treatment of engineer capability, SEAC provides a powerful tool to examine sustainment engineering planning.

13. Future. ESC has called SEAC a prototype engineer model. It should not be inferred that the model is not operational; it was successfully used on the EAK study. It is not, however, intended to be immutable. ESC studies different OPLANs and different elements of COMMZ engineering missions. The following items represent a list of features that can be envisioned for the model. The object-based structure of SEAC enables changes and, in particular, additions to the model to be accomplished easier than in other procedural languages.

a. **Class IV materiel**. Associated with CATCODE construction and repair tasks are lists of the materiel needed to complete the job. It would be relatively easy to include such information in the model and indicate how such supplies were consumed, by time, and by location. And, analogous to personnel capability checking, SEAC could compare on-hand assets with required amounts to identify where shortages hindered completion.

b. **Maintenance**. The major reason why maintenance is not presently in the model was data. The structure of the model would easily accommodate such work without significant changes to existing codes. The only obstacle is credible data to base workload.

c. **Logistic network**. SEAC does not contain an explicit representation of the loadings and movement of supplies over the network. It would not be hard, however, to add the structure and logic necessary to implement the depot system.

d. **Equipment**. It would be extremely useful to be able to compare a unit's equipment with its workload requirements. Unfortunately, equipment amounts associated with CATCODE items are meager and unreliable. If they were available, equipment would likely be handled much the same as manpower capability is treated.

e. **Special missions**. The special skills, equipment, and locations for work performed by pipeline, port, and other engineer units cannot be done

by just any unit. It would be interesting to expand the task and the unit treatment associated with these activities. As it is now, these units are frequently included in TPFDD according to rules of allocation, not verifiable need.

f. **Interruptability.** SEAC makes hard-wired decisions about resource allocation. Whether it is where engineers shall work, or where enemy strikes will occur, the model will react in a well understood way. It might be useful and would not be difficult to insert a human player into the decision process. Normally, the sheer number of such decisions might argue against adding this feature. Because of SEAC's rich and detailed representation of the COMMZ, however, the interrupts could be confined to queries from a single installation, or a single region.

g. **Effects.** One of the difficulties of modeling engineers is estimating what the effect of their efforts has on the rest of the force. Units say they need facilities. The assumption is that if they don't have them, the war will not go as well. Now it is impossible to say that construction of a damaged pier will mean we will win two brigade-sized engagements. But we might be able to determine how much the port's processing capability is reduced. Such results could then be used in theater-level war games to better represent engineer contributions.

VI. POSTSCRIPT

14. Object-oriented Programming (OOP). In addition to being successfully used as the basis for assessing an actual OPLAN, SEAC is important as an example of object-oriented programming. A popular theme today in many semi-technical articles laments the marginal productivity gains experienced in software development in the face of successive orders of magnitude improvements in hardware capability. Sure-fire remedies include: artificial intelligence methods, ADA, fourth generation languages, OOP, etc. Although each solution holds promise, there are few examples of successful working systems. Interestingly, ESC was initially attracted to SIMULA because it offered the process or scenario view of simulation.²⁰ It did not take long, however, before the versatility and power of the class and inheritance features began to dominate the design efforts. The modularity hoped for but never quite realized in languages, such as FORTRAN, evolved naturally in the code written in SIMULA. The experience indicated that translating a modeling problem into computer code appears to be much easier using object-oriented design techniques. It can simply not be overstated that the notational and representational attributes of computer languages are not just a means to instruct a computer, but are cognitive devices that assist the developer in structuring and abstracting a problem. SIMULA clearly proved superior in the area of discrete event modeling to the alternative at ESC -- FORTRAN 77. Apart from the strong simulation context provided in the SIMULA, other military modelers may find three aspects of SEAC's development to be particularly interesting: data definition, compile time checking, and inter-object communication.

a. **Data structuring.** The power of object-oriented languages is in their ability to encapsulate data structure and behavior. FORTRAN provides

²⁰SIMULA 67 is generally considered to be the first object-oriented language. There is a great deal of discussion today over those characteristics of truly object-oriented languages. They should have class templates for objects. They should permit inheritance (is multiple better than single?). Should messages be the only means of inter-object communications? Should they employ dynamic versus static binding? Perhaps the purists will sort these questions out. As far as ESC is concerned, SIMULA provided a powerful vehicle to design and implement a system that it felt employed object-oriented programming techniques.

integers, reals, and text types, and recognizes arithmetic operations and alphanumeric manipulation and checking. In SIMULA one can create a type for each element of the problem (SEAC defined types for facilities, installations, planes, etc.). Moreover, the various procedures associated with changing and manipulating these elements and their attributes can be defined. The focus on data identification, structuring, and associations, coupled with localizing operations within objects that modify these data, is the heart of object-oriented design. In particular, the inheritance feature of object-oriented design, i.e., the ability to define new object classes from existing classes, probably comes closer to making reusable code a reality than traditional subroutine libraries, or even ADA's packages and Modula's modules.²¹

b. **Compilation.** One of the interesting artifacts of object-oriented environments is that some, particularly if LISP-based, have arisen as interpreted languages, while others have evolved as compiled languages. Interpreted environments frequently are more user friendly, and do not necessarily type objects until the last possible moment. Compiled languages usually run faster, but will be stricter in terms of the mechanics of separate compilations and the specification of reference variables. Another distinguishing characteristic is referred to as strong typing. Strong typing, within and between compiled units, results in many unintended errors being flagged at compile-time. Finding these errors at execution time can be extremely difficult, or worse still, they may exist but remain undiscovered. ESC's overall experience with SIMULA, a compiled language, was extremely favorable and in the large model environment of the military, may provide the better approach.

c. **Communication.** Another important area of software development is data hiding (sometimes also referred to as scoping).²² This refers to the ability of one program unit to modify another unit's data. Intentional or unintentional changes of data have been the downfall of many programs because data hiding could not be enforced. Object-oriented languages almost by definition include some form of data hiding. Some, the most extreme being

²¹Cox, Brad J., *Object Oriented Programming -- An Evolutionary Approach* (Addison-Wesley, August 1986).

²²Wiener, Richard and Richard Sinovec, *Software Engineering with Modula-2 and ADA* (John Wiley & Sons, 1984) pp. 64-68.

Smalltalk, limit the accessibility to invoking methods or passing "messages" between objects.²³ SIMULA's data accessibility spans many possibilities: as an ALGOL superset, it exhibits the normal global/local scoping rules for block-oriented languages; object parameters and outer-block specified variables are accessible unless declared to be protected; variables that are inherited by subclass objects can be hidden from those objects. Unfortunately, the version of SIMULA used by ESC did not offer hidden/protected specifications. There was no way, other than by convention, to limit the permissible direct access of another object's data as long as the type and qualification of all variable labels were correct (SIMULA uses dot notation to designate object parameters). The plus side of such access, however, is the inherent improvement in execution speed of function or reference access over strict message passing communication. (Message overhead can be 2 to 2.5 times that of a simple function call).²⁴ Also, such access within SEAC was largely limited to interrogating the variables; changes and updates to the variables were largely accomplished by invoking procedures or methods that were part of the effected objects.

LAST PAGE OF MAIN PAPER

²³*Smalltalk/V* (Digitalk Inc., August 1987)

²⁴Cox, p. 134.

ANNEX A

SIMULA

ANNEX A

SIMULA

<u>Paragraph</u>		<u>Page</u>
1	Purpose	A-1
2	Scope	A-1
3	What's in a Language?	A-2
4	What is SIMULA?	A-2
5	Major Features	A-3
6	SIMULA And Other HOLs	A-10
7	The Compiler	A-11

Figure

A-1	Converting Ideas Into a Model	A-3
A-2	Basic CLASS Declaration Format	A-4
A-3	Example of a Simple Class	A-5
A-4	General CLASS Format	A-6
A-5	EXAMPLE: Definitions of CLASSES A & B	A-6
A-6	EXAMPLE: Equivalent Definition of CLASS B	A-7
A-7	Example of SIMULA's Virtual Declaration	A-9

1. Purpose. This annex is a brief introduction to SIMULA, the computer language in which the SEAC model is written.

2. Scope. This annex is not meant to be a tutorial or reference guide; instead, it will highlight several major features of SIMULA that play an important part in the design of SEAC. Its goal is to introduce the reader to the general structure and features of the language as an aid in understanding the program listings and discussions found in Annex B, Program Documentation. For any one who desires to learn more about the language, several books and manuals are available.^{1,2}

¹Dahl, O. J., Myhrhaug, B., and Nygaard, K., *SIMULA 67 Common Base Language* (Norwegian Computing Center, October 1970).

²Lamprecht, Gunther, *Introduction to SIMULA 67* (Friedr. Vieweg & Sohn, 1983).

3. What's in a Language? Just as a person's ability to express ideas is dependent upon his language skills (vocabulary and grammar), a computer modeler is similarly dependent upon the computer language that he uses and his facility with it (commands and syntax). How the model is constructed is inextricably tied to the language that translates the modeler's abstractions into executable computer instructions. Computer languages in general, and SIMULA in particular, influence the way a programmer thinks about a problem. An idea or operation represented in a computer program can probably be expressed in several computer languages. The language choice, however, will influence the length of time necessary to reduce that idea to correct executable code. The features of SIMULA enable a modeler to spend more time on conceptualization; SIMULA provides the expressive framework and the compiler handles the translation into machine code. Figure A-1 suggests the perceived advantages in using SIMULA to represent abstractions. With the language playing such an important role, to understand the design of SEAC (a rather complex model), one must therefore be acquainted with the basic constructs of SIMULA.

4. What is SIMULA? The SIMulation Language (SIMULA) was initially developed at the Norwegian Computing Center in Oslo as a simulation language that could also be used for system descriptions. In fact, early users considered the communicative value of the language to be as important as its executability. Many people interested in programming languages have heard of SIMULA, but few have had any opportunity to become acquainted with its features. SEAC actually employs the second version of the language. While the first version achieved its express purpose, the prescience of the designers spurred them on to revise the initial design and to make SIMULA a true, general-purpose language. (The story of SIMULA's development is recounted in an interesting collection of language histories compiled by a major computer association.)³ While SIMULA has existed and been available for almost 20 years, it has never gained a following in the United States. That SIMULA was better received in the European community than in the US, was no doubt because of the ALGOL orientation in Europe, and the FORTRAN mindset here. It is only

³Nygaard, K. and Dahl, O.J., *History of Programming Languages*, Part IX, "The Development of the SIMULA Languages", (Academic Press, 1981).

CONVERTING IDEAS INTO A MODEL

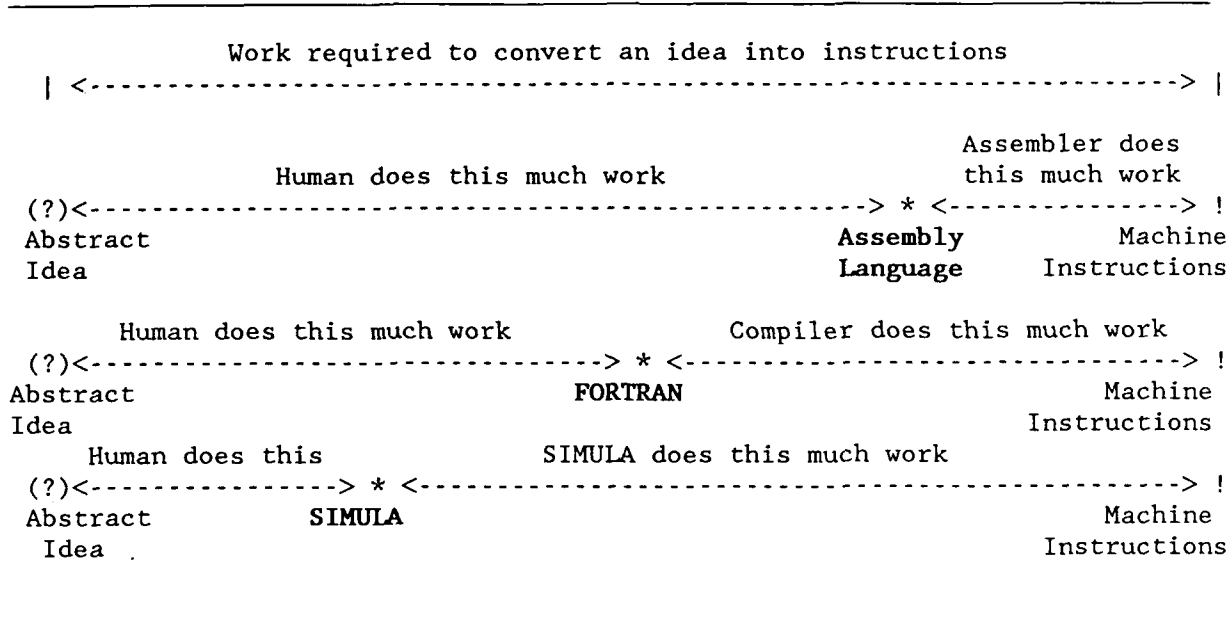


Figure A-1

recently that SIMULA has received the recognition it deserved due, in great part, to a better understanding of what software engineering research indicates a computer language should provide.

5. Major Features. SIMULA is an extremely powerful general purpose computer language. Its CLASS construct was the seminal development in what is now called object-oriented programming (see paragraph 5.b). It supports the software engineering goals of abstraction, information hiding, modularity, and localization. The language designers also sought a language that could be used by others to easily implement their own tools or constructs and in a way extend the language toward specialized applications. Understanding the power and features of SIMULA is not an easy undertaking. This section, as stated above, is not meant to be a tutorial but seeks only to introduce several major features of SIMULA that were relied upon during the development of SEAC.

a. ALGOL Kernel. SIMULA is an extension of the computer language ALGOL. SIMULA adopts the form of ALGOL and most of its rules and constructs. One of the often stated strengths of SIMULA is that it has the rich algorithmic and procedural power of ALGOL to manipulate information (unlike some primarily simulation languages such as GPSS). SIMULA, however, extends ALGOL's block

concept to allow the generation of block instances that are conceptually separate from the enclosing block, as well as facilities for the treatment of quasi-parallel systems.

b. CLASSES. The SIMULA CLASS is a generalization of an ALGOL block, and was the first implementation of a software construct presently called "data abstraction." The importance of this software device for problem decomposition and program construction was initially underestimated, even by SIMULA's developers, and has only become clear in recent years. A data abstraction defines a data object and the operations and algorithms that are associated with it. The use of objects (which has given rise to the term object-oriented programming), which are the instantiations of abstract data types has, in fact, become one of the pillars of current software engineering theory.

(1) Declaration. A CLASS declaration has the following form:

BASIC CLASS DECLARATION FORMAT

```
CLASS class name (formal parameters);  
parameter specifications;  
BEGIN  
attribute declarations;  
executable statements;  
END;
```

Figure A-2

Interestingly, most of the items are optional. That is, a CLASS need not have parameters, attributes (procedures or variables), or executable statements. Of course the presence of formal parameters requires accompanying specifications, and statements would generally use either global variables, parameters, or attributes that have been defined. A CLASS is actually only a template. When a CLASS is invoked, an object is created that takes the form the CLASS defines. Multiple copies of the same or different CLASSES may coexist in memory at the same time. This occurs typically when the following expression is encountered somewhere in a program:

NEW class-name (actual parameters)

This tells the execution supervisor program that an object, defined by the specifications for CLASS "class-name," is to be created with the parameters provided. It is only at that time that memory is allocated and execution of statements begins (if they exist). Figure A-3 shows a small CLASS, A, which has a single parameter, P, an internal procedure, PRINT, and a CLASS body which first modifies the parameter and then invokes PRINT. Execution of this program would produce this line of output:

5 500

EXAMPLE OF A SIMPLE CLASS

```

BEGIN
CLASS A(P);
  INTEGER P;
  BEGIN
    INTEGER N;
    PROCEDURE PRINT;
      BEGIN
        OUTINT(P,5); OUTINT(N,5); OUTIMAGE;
      END

    N := 100 * P;
    PRINT;
  END -- CLASS A --;

NEW A(5);
END ** PROGRAM **;
```

Figure A-3

(2) Concatenation. The above example of a CLASS is not all that different from a FORTRAN subroutine that takes a number, modifies it, and prints it. In Figure A-4 the power of the CLASS becomes more evident when the general format of a CLASS is understood. The key entry is the inclusion of the prefix to CLASS S. This is to be interpreted to mean that a CLASS C has been defined elsewhere. By prefixing the definition of CLASS S with C, a composite CLASS is defined which is the concatenation of both CLASS definitions. In SIMULA class S would be called a subCLASS of CLASS C and would inherit the attributes, procedures, and body of the prefix CLASS. For example suppose CLASSES A and B were defined as in Figure A-5.

GENERAL CLASS FORMAT

```
C CLASS S(p);  
  s;  
  BEGIN  
  d;  
  e1;  
  INNER;  
  e2;  
  END;
```

where:

C is the prefix CLASS (optional)
S is the CLASS being defined
p are the formal parameters associated with S
s are the specifications for the formal parameters
d are the attribute declarations associated with S
e1 and e2 are executable statements
(INNER will be explained below)

Figure A-4

EXAMPLE: DEFINITIONS OF CLASSES A & B

CLASS A(Ap);	A CLASS B(Bp);
As;	Bs;
BEGIN	BEGIN
Ad;	Bd;
Ae1;	Be1;
INNER;	INNER;
Ae2;	Be2;
END;	END;

Figure A-5

The preceding definition of B is actually equivalent to the CLASS defined in Figure A-6.

EXAMPLE: EQUIVALENT DEFINITION OF CLASS B

```
CLASS B(Ap,Bp);  
  As;Bs;  
  BEGIN  
    Ad;Bd;  
    Ae1;  
      Bel;  
  INNER;  
    Be2;  
    Ae2;  
  END;
```

Figure A-6

The specification of C CLASS S thus defines a compound object which merges the elements of both CLASS definitions. It may now become apparent what INNER means; it provides an opportunity to indicate where the executable statements of any subCLASS would be executed. If INNER is not expressly included, then it is assumed to be at the end of the executable statements in the CLASS. For those readers interested in prototyping and top-down program design, the implication of CLASS concatenation should be apparent.

c. **VIRTUAL Procedures.** The objective of the software design process is to prepare a coherent and valid representation of the system. Present design methodologies generally support a top-down approach, where a system is decomposed in stages. At the highest level one need not worry about implementation details, but rather only concentrate on identifying the overall processes or components that should be included in the system. At each level the process is further refined. Finally, after the overall structure of the system is well defined, the operational details are implemented. SIMULA provides a powerful tool to assist system designers -- VIRTUAL Quantities. This is a means to permit procedure attribute redeclaration at one prefix level that is valid at outer prefix levels. Variable "x" is defined in the main program as a reference variable for CLASS c1. CLASSES c2 and c3 are defined as

subCLASSES of c1 and c2 respectively. Note that two attributes are declared in each CLASS declaration: an integer variable "i" and a procedure "sub." Normal scoping rules for SIMULA specify that CLASS variables are normally accessible only to variables with matching reference qualifications. Thus variable reference x.i and procedure invocation x.sub should access attributes defined in c1, since that is the qualification of x. The VIRTUAL declaration extends scoping to permit redefinition (or definition) of procedures that are specified in subclasses. Thus, in the case of the program above, x is qualified to permit referencing objects from CLASS c1. In lines 31, 32, and 33 of Figure A-7, x points to objects of CLASSES c1, c2, and c3 respectively. When x.i is printed for each object reference, it accesses the i variable defined in CLASS c1. Because of the VIRTUAL declaration for procedure sub, however, each call invokes the routine defined at the lowest CLASS in the concatenated CLASSES: c1, clc2, or clc2c3. (An example of VIRTUAL procedures in SEAC is the REPAIR function associated with facility objects. The repair of runway craters, storage tanks, and maintenance shops entails different treatment. VIRTUAL-izing the REPAIR procedure permits SEAC to apply the correct repair approach without having to identify which FACILITY sub-subCLASS has suffered damage.) SIMULA's run time system will ensure that the correct procedure is used.

d. **Separate Compilations.** Software engineering practice emphasizes designs that partition systems or modules along sound functional lines.⁴ Also important to promoting productivity is the ability to accumulate a library of general or special application routines that can be used in new programs. SIMULA provides such facilities by supporting separate compilation. Each "module" is defined and compiled as a separate external CLASS (or procedure). When needed, the CLASS is declared in an EXTERNAL statement in the target program (or another "module"), and its attributes are accessed by prefixing it to either a block or another CLASS. (The compiler will search through the file directories for the needed CLASS.) The implications are clear; code implementing complex arithmetic, matrix operations, special data structures, specific problem-oriented code, etc., can be separately developed but conveniently reused in application programs.

⁴Parnas, David, On the Criteria to be Used in Decomposing Systems into Modules, 15 Communications of the Association for Computing Machinery 12, 1053 (December 1972).

EXAMPLE OF SIMULA's VIRTUAL DECLARATION

I. Program

```
1:  COMMENT This program illustrates the VIRTUAL concept;
2:
3:  BEGIN
4:  REF(c1) x;
5:
6:  CLASS c1;
7:    VIRTUAL: PROCEDURE sub;      ! <-----Virtual Procedure Defined;
8:    BEGIN
9:    INTEGER i;
10:   PROCEDURE sub;
11:     BEGIN OUTTEXT("  sub/c1");OUTIMAGE; END;
12:   i := 111;
13:   END - c1 -;
14:
15: c1 CLASS c2;
16:   BEGIN
17:   INTEGER i;
18:   PROCEDURE sub;
19:     BEGIN OUTTEXT("    sub/c2");OUTIMAGE; END;
20:   i := 222;
21:   END - c2 -;
22:
23: c2 CLASS c3;
24:   BEGIN
25:   INTEGER i;
26:   PROCEDURE sub;
27:     BEGIN OUTTEXT("      sub/c3");OUTIMAGE; END;
28:   i := 333;
29:   END - c3 -;
30:
31: x :- NEW c1; OUTTEXT(" x (c1/c1).. i =");OUTINT(x.i,4); x.sub;
32: x :- NEW c2; OUTTEXT(" x (c1/c2).. i =");OUTINT(x.i,4); x.sub;
33: x :- NEW c3; OUTTEXT(" x (c1/c3).. i =");OUTINT(x.i,4); x.sub;
34:
35: END .. PROGRAM ..;
```

II. Program Results

```
x (c1/c1).. i = 111  sub/c1
x (c1/c2).. i = 111  sub/c2
x (c1/c3).. i = 111  sub/c3
```

Figure A-7

6. SIMULA And Other HOLs. The reader may find it interesting that a language exhibiting many of the traits now being touted for recently developed languages has been around for almost twenty years. Just how well does SIMULA compare to languages that have been expressly designed to satisfy the needs of software engineering? Currently in Department of the Army, FORTRAN and SIMSCRIPT II.5 have been used to implement a preponderance of existing models, with the latter more common among more recently developed models, especially those in the Army's wargame hierarchy. One need only say a few things about FORTRAN: it is by far the most widely used language for modeling; the majority of its users seem firmly committed to continued use of it; with millions of lines of code invested in existing systems, it will be with us for many years to come; and finally, it is deficient in the tools needed for good software engineering practice. SIMSCRIPT is a language that is frequently compared to SIMULA when comparing simulation languages. While SIMSCRIPT is a good simulation language and much preferred to FORTRAN, it does not offer much more than FORTRAN does in the area of software engineering (e.g., compile time checking characteristic of the strongly-typed languages such as ADA and SIMULA). The two languages that seem more appropriate to contrast with SIMULA as a high order language are ADA⁵ and Modula-2⁶. (PASCAL is not discussed because many of its flaws have been remedied by its developer in Modula-2.) ADA was the result of a determination that DOD software costs were escalating at an alarming rate, that things were only getting worse, and that hardware and software systems performance would suffer as a result. A group was formed to study the problem and offer solutions. ADA is the result of its recommendation to develop a new language that would be at home in either a real-time, multiprocessor or mainframe environment. In fact, ADA has been declared as the language of choice for all embedded software. Modula-2, on the other hand, is the product of one man and evinces its character as single-station, single-user language. It would take many pages to discuss and contrast the features of ADA, SIMULA, and Modula-2. The remarkable thing is, however, that SIMULA's CLASS embodies in a single concept ADA generics, tasks, data types or packages,

⁵ANSI/MIL-STD-1815A-1983, *Reference Manual for the Ada Programming Language*, (American National Standards, Inc., 1983).

⁶Wirth, Niklaus, *Programming in Modula-2* (Springer-Verlag, Berlin Heidelberg, New York, 1982).

and Modula-2 records, processes, or modules. Admittedly the embodiment may not be an exact functional duplication (sometimes it's better, sometimes it's not). Nonetheless, the CLASS must be recognized for its power and its elegant simplicity. It is puzzling that SIMULA never has received wider attention; perhaps it was (and still is) a case of being ahead of its time.

7. The Compiler. SIMULA, as a strongly typed language, checks every program expression to identify typing inconsistencies and illegal references that might exist in the code. A variable's "type" is its specification label. Variables "speed" and "pages" might have types of real and integer respectively, while variable "house" might be a reference type for CLASS "buildings". The compiler makes sure that assignments are consistent, so a TEXT variable is not set equal to an integer value. It thus finds many errors that a FORTRAN compiler for instance would not detect. Identifying errors at compile time rather than run time is extremely advantageous. Run time (i.e., execution) errors are much more difficult to detect, duplicate, and trace. The benefit has a price -- longer compilation times. Fortunately SIMULA supports separate compilation. In fact, because of the slowness of the compiler, it becomes almost compulsory to break the model into logical modules to reduce compilation turnaround. It takes almost 100 PR1ME 750 cpu minutes to compile all the SEAC programs (using version 1 of PR1ME's compiler). Wall time is even worse since it is a function of the number of other user processes competing with the compiler for machine resources. When usage is low, cpu time approximates wall time. When the machine is being heavily used, however, wall time frequently becomes 3 or 4 times greater than cpu time. ESC has never been happy with the compilation times, but can rationalize it as the price for dealing with a strongly typed system and the compile-time error identification that it provides.

ANNEX B

DETAILED SYSTEM DESCRIPTIONS

Annex B

Detailed System Descriptions

<u>Paragraph</u>	<u>Page</u>
Introduction	B-7
Scope	B-7
Special Features	B-7
SIMULA's CLASS SIMULATION	B-7
Model	B-8
Class Description Format	B-13
Module Descriptions	B-14
Module AVLTREE	B-14
Module DATAI	B-15
Module DATA2	B-18
Module PROCS	B-20
Module THREAT	B-21
Module SEAKM	B-22
Class Descriptions	B-24
Class COMPONENT	B-24
Class ENGRPOOL	B-25
Class FACELEM	B-26
Class MAPCELL	B-27
Class NKBASES	B-31
Class POLICY	B-33
Class REPORTER	B-34
Class REPORTS	B-36
Class ROOT	B-39
Class CATREE	B-41
Class FACTREE	B-43
Class ECAPTREE	B-47
Class UOMTREE	B-48
Class GEOTREE	B-50
Class TGTPREF	B-52
Class LINKAGE	B-53
Class HEAD	B-54
Class ORDLIST	B-55
Class LINK	B-56
Class NODE	B-57
Class CATCOD	B-61
Class FACTOR	B-63
Class FACTORX	B-65
Class ENGRCAP	B-67
Class UNOFMS	B-68
Class COMPO	B-69
Class GEOLOC	B-70
Class GEOSUB	B-72

Class PROPERTY	B-73
Class ENGRUNIT	B-78
Class ORDNANCE	B-80
Class BOMB	B-81
Class ROCKET	B-82
Class DEMOL	B-83
Class AIRCRAFT	B-84
Class MUNITION	B-85
Class PLANE	B-86
Class NKAIRBASE	B-89
Class SPFUNIT	B-91
Class TEAMORG	B-92
Class FACILITY	B-95
Class SURFACE	B-98
Class RUNWAY	B-101
Class PAVEMENT	B-101
Class HARDSTAND	B-101
Class BUILDING	B-101
Class COMO	B-103
Class OPNS	B-104
Class REVETMENT	B-104
Class PIER	B-104
Class SHOP	B-104
Class STORAGE	B-105
Class MEDICAL	B-105
Class ADMIN	B-105
Class QTRS	B-106
Class PETRO	B-106
Class POLFACIL	B-107
Class TANK	B-107
Class PIPE	B-108
Class UTILITY	B-109
Class POWER	B-112
Class WASTE	B-112
Class WATER	B-112
Class TRANSPD	B-113
Class ROAD	B-115
Class RR	B-115
Class AUNIT	B-115
Class REGION	B-116
Class REGCELL	B-126
Class HITS	B-127
Class TASK	B-128
Class INSTALLATION	B-131
Class AIRBASE	B-138
Class AIRPOD	B-141
Class CAMP	B-142
Class CENTER	B-143
Class PORT	B-145
Class SEAPOD	B-146
Class ADA	B-147
Class HOSPITAL	B-148
Class DEPOT	B-148
Class ELEC	B-148

Processes	B-149
Class RSD	B-149
Class CHANGE	B-153
Class MISSION	B-158
Class MISGEN	B-161
Class REARAXN	B-165
Procedures -- Functions	B-170
Procedure RANDPICK	B-171
Procedure CATOFAC	B-172
Procedure PRIORANK	B-174
Procedure RATE	B-175
Procedure COUNCK	B-176
Procedure C	B-177
Procedure SVCSCK	B-178
Procedure DIARY	B-179
Procedure GET	B-179
Procedure TYPEINSTALL	B-180
Procedure DEPLOY	B-182
Procedure BLAST	B-182
Procedures -- Major	B-186
Procedure BUILDMAP	B-186
Procedure ADDASSETS	B-191
Procedure SETUP	B-194
Procedure WORK	B-199
Index	B-202

Figure

B-1	SEAC Program Modules	B-10
B-2	Class Hierarchy	B-11
B-3	General Format of Class Variable/Parameter Tables	B-13
B-4	Declarations and Specifications for Module AVLTREE	B-15
B-5	Variable Parameters and Attributes for Module AVLTREE	B-15
B-6	Top-level Declarations and Specifications for Module DATAI	B-16
B-7	Variable Parameters and Attributes for Module DATAI	B-17
B-8	Top-level Declarations and Specifications for Module DATA2	B-18
B-9	Variable Parameters and Attributes for Module DATAI	B-19
B-10	Top-level Declarations and Specifications for Module PROCS	B-20
B-11	Variable Parameters and Attributes for Module PROCS	B-21
B-12	Top-level Declarations and Specifications for Module Threat	B-21
B-13	Variable Parameters and Attributes for Module THREAT	B-22
B-14	Top-level Declarations and Specifications for SEAKM	B-23
B-15	Variable Parameters and Attributes for Module SEAKM	B-23
B-16	Parameters and Attributes for Class COMPONENT	B-25
B-17	Listing of Class COMPONENT	B-25
B-18	Parameters and Attributes for Class ENGRPOOL	B-26
B-19	Listing of Class ENGRPOOL	B-26
B-20	Parameters and Attributes for Class FACELEM	B-26
B-21	Listing of Class FACELEM	B-26
B-22	Parameters and Attributes for Class MAPCELL	B-28

B-23	Listing of Class MAPCELL	B-29
B-24	Parameters and Attributes for Class NKBASES	B-31
B-25	Listing of Class NKBASES	B-32
B-26	Parameters and Attributes for Class POLICY	B-33
B-27	Listing of Class POLICY	B-34
B-28	Parameters and Attributes for Class REPORTER	B-35
B-29	Listing of Class REPORTER	B-36
B-30	Parameters and Attributes for Class REPORTS	B-37
B-31	Listing of Class REPORTS	B-37
B-32	Parameters and Attributes for Class ROOT	B-40
B-33	Listing of Class ROOT	B-41
B-34	Parameters and Attributes for Class CATREE	B-41
B-35	Listing of Class CATREE	B-42
B-36	Parameters and Attributes for Class FACTREE	B-44
B-37	Listing of Class FACTREE	B-44
B-38	Parameters and Attributes for Class ECAPTREE	B-47
B-39	Listing of Class ECAPTREE	B-47
B-40	Parameters and Attributes for Class UOMTREE	B-49
B-41	Listing of Class UOMTREE	B-49
B-42	Parameters and Attributes for Class GEOTREE	B-50
B-43	Listing of Class GEOTREE	B-51
B-44	Parameters and Attributes for Class TGTPREF	B-52
B-45	Listing of Class TGTPREF	B-53
B-46	Parameters and Attributes for Class LINKAGE	B-53
B-47	Listing of Class LINKAGE	B-54
B-48	Parameters and Attributes for Class HEAD	B-54
B-49	Listing of Class HEAD	B-55
B-50	Parameters and Attributes for Class ORDLIST	B-55
B-51	Listing of Class ORDLIST	B-56
B-52	Parameters and Attributes for Class LINK	B-57
B-53	Listing of Class LINK	B-57
B-54	Parameters and Attributes for Class NODE	B-58
B-55	Listing of Class NODE	B-59
B-56	Parameters and Attributes for Class CATCOD	B-62
B-57	Listing of Class CATCOD	B-63
B-58	Parameters and Attributes for Class FACTOR	B-64
B-59	Listing of Class FACTOR	B-65
B-60	Parameters and Attributes for Class FACTORX	B-66
B-61	Listing of Class FACTORX	B-66
B-62	Parameters and Attributes for Class ENGRCAP	B-67
B-63	Listing of Class ENGRCAP	B-68
B-64	Parameters and Attributes for Class UNOFMS	B-69
B-65	Listing of Class UNOFMS	B-69
B-66	Parameters and Attributes for Class COMPO	B-70
B-67	Listing of Class COMPO	B-70
B-68	Parameters and Attributes for Class GEOLOC	B-71
B-69	Listing of Class GEOLOC	B-72
B-70	Parameters and Attributes for Class GEOSUB	B-73
B-71	Listing of Class GEOSUB	B-73
B-72	Parameters and Attributes for Class PROPERTY	B-74
B-73	Listing of Class PROPERTY	B-75
B-74	Parameters and Attributes for Class ENGRUNIT	B-79
B-75	Listing of Class ENGRUNIT	B-80
B-76	Parameters and Attributes for Class ORDNANCE	B-81

B-77	Listing of Class ORDNANCE	B-81
B-78	Parameters and Attributes for Class BOMB	B-82
B-79	Listing of Class BOMB	B-82
B-80	Parameters and Attributes for Class ROCKET	B-83
B-81	Listing of Class ROCKET	B-83
B-82	Parameters and Attributes for Class DEMOL	B-84
B-83	Listing of Class DEMOL	B-84
B-84	Parameters and Attributes for Class AIRCRAFT	B-85
B-85	Listing of Class AIRCRAFT	B-85
B-86	Parameters and Attributes for Class MUNITION	B-86
B-87	Listing of Class MUNITION	B-86
B-88	Parameters and Attributes for Class PLANE	B-87
B-89	Listing of Class PLANE	B-87
B-90	Parameters and Attributes for Class NKAIRBASE	B-90
B-91	Listing of Class NKAIRBASE	B-90
B-92	Parameters and Attributes for Class SPFUNIT	B-92
B-93	Listing of Class SPFUNIT	B-92
B-94	Parameters and Attributes for Class TEAMORG	B-93
B-95	Listing of Class TEAMORG	B-93
B-96	Parameters and Attributes for Class FACILITY	B-96
B-97	Listing of Class FACILITY	B-97
B-98	Parameters and Attributes for Class SURFACE	B-99
B-99	Listing of Class SURFACE	B-99
B-100	Listing of Class RUNWAY	B-101
B-101	Listing of Class PAVEMENT	B-101
B-102	Listing of Class HARDSTAND	B-101
B-103	Parameters and Attributes for Class BUILDING	B-102
B-104	Listing of Class BUILDING	B-102
B-105	Listing of Class COMO	B-103
B-106	Listing of Class OPNS	B-104
B-107	Listing of Class REVETMENT	B-104
B-108	Listing of Class PIER	B-104
B-109	Listing of Class SHOP	B-105
B-110	Listing of Class STORAGE	B-105
B-111	Listing of Class MEDICAL	B-105
B-112	Listing of Class ADMIN	B-106
B-113	Listing of Class QTRS	B-106
B-114	Parameters and Attributes for Class PETRO	B-106
B-115	Listing of Class PETRO	B-107
B-116	Listing of Class POLFACIL	B-107
B-117	Listing of Class TANK	B-108
B-118	Listing of Class PIPE	B-109
B-119	Parameters and Attributes for Class UTILITY	B-110
B-120	Listing of Class UTILITY	B-110
B-121	Listing of Class POWER	B-112
B-122	Listing of Class WASTE	B-112
B-123	Listing of Class WATER	B-113
B-124	Parameters and Attributes for Class TRANSP	B-113
B-125	Listing of Class TRANSP	B-114
B-126	Listing of Class ROAD	B-115
B-127	Listing of Class RR	B-115
B-128	Parameters and Attributes for Class AUNIT	B-116
B-129	Listing of Class AUNIT	B-116
B-130	Parameters and Attributes for Class REGION	B-118

B-131 Listing of Class REGION	B-119
B-132 Parameters and Attributes for Class REGCELL	B-126
B-133 Listing of Class REGCELL	B-127
B-134 Parameters and Attributes for Class HITS	B-127
B-135 Listing of Class HITS	B-128
B-136 Parameters and Attributes for Class TASK	B-129
B-137 Listing of Class TASK	B-129
B-138 Parameters and Attributes for Class INSTALLATION	B-131
B-139 Listing of Class INSTALLATION	B-133
B-140 Parameters and Attributes for Class AIRBASE	B-139
B-141 Listing of Class AIRBASE	B-139
B-142 Parameters and Attributes for Class AIRPOD	B-141
B-143 Listing of Class AIRPOD	B-141
B-144 Parameters and Attributes for Class CAMP	B-142
B-145 Listing of Class CAMP	B-142
B-146 Parameters and Attributes for Class CENTER	B-143
B-147 Listing of Class CENTER	B-144
B-148 Parameters and Attributes for Class PORT	B-146
B-149 Listing of Class PORT	B-146
B-150 Parameters and Attributes for Class SEAPOD	B-146
B-151 Listing of Class SEAPOD	B-147
B-152 Listing of Class ADA	B-148
B-153 Listing of Class HOSPITAL	B-148
B-154 Listing of Class DEPOT	B-148
B-155 Listing of Class ELEC	B-148
B-156 Parameters and Attributes for Class RSD	B-150
B-157 Listing of Class RSD	B-151
B-158 Parameters and Attributes for Class CHANGE	B-153
B-159 Listing of Class CHANGE	B-155
B-160 Parameters and Attributes for Class MISSION	B-159
B-161 Listing of Class MISSION	B-159
B-162 Parameters and Attributes for Class MISGEN	B-162
B-163 Listing of Class MISGEN	B-163
B-164 Parameters and Attributes for Class REARAXN	B-166
B-165 Listing of Class REARAXN	B-167
B-166 Parameters and Local Variables for Procedure RANDPICK	B-171
B-167 Listing of Procedure RANDPICK	B-172
B-168 Parameters and Local Variables for Procedure CATOFAC	B-172
B-169 Listing of Procedure CATOFAC	B-173
B-170 Parameters and Local Variables for Procedure PRIORANK	B-175
B-171 Listing of Procedure PRIORANK	B-175
B-172 Parameters and Local Variables for Procedure RATE	B-176
B-173 Listing of Procedure RATE	B-176
B-174 Parameters and Local Variables for Procedure COUNCK	B-177
B-175 Listing of Procedure COUNCK	B-177
B-176 Parameters and Local Variables for Procedure C	B-178
B-177 Listing of Procedure C	B-178
B-178 Parameters and Local Variables for Procedure SVCSCCK	B-178
B-179 Listing of Procedure SVCSCCK	B-179
B-180 Parameters and Local Variables for Procedure DIARY	B-179
B-181 Listing of Procedure DIARY	B-179
B-182 Parameters and Local Variables for Procedure GET	B-180
B-183 Listing of Procedure GET	B-180
B-184 Parameters and Local Variables for Procedure TYPEINSTALL	B-181

B-185 Listing of Procedure TYPEINSTALL	B-181
B-186 Parameters and Local Variables for Procedure DEPLOY	B-182
B-187 Listing of Procedure DEPLOY	B-182
B-188 Parameters and Local Variables for Procedure BLAST	B-183
B-189 Listing of Procedure BLAST	B-184
B-190 Parameters and Local Variables for Procedure BUILDMAP	B-187
B-191 Listing of Procedure BUILDMAP	B-188
B-192 Parameters and Local Variables for Procedure ADDASSETS	B-192
B-193 Listing of Procedure ADDASSETS	B-192
B-194 Parameters and Local Variables for Procedure SETUP	B-195
B-195 Listing of Procedure SETUP	B-196
B-196 Parameters and Local Variables for Procedure WORK	B-199
B-197 Listing of Procedure WORK	B-200

1. Introduction. This annex presents the SEAC computer code. In addition to presenting the individual components of the model, the overall structure of the model is also introduced.

2. Scope. SEAC was developed to be an in-house tool with which ESC could conduct a variety of analyses concerning COMMZ and RCZ engineer related problems. It was intended, however, to be a prototype in that even though it has demonstrated its viability as a model, succeeding uses will probably require that parts of the model be enhanced or changed, or that new features be added. Rather than being a black box, SEAC provides a study environment in which different problems may be analyzed. Making such changes necessarily requires understanding the code and the language in which SEAC is written. Thus the reason for this manual. Rather than exhaustively tracing the logic of each program unit, line by line, ESC has chosen to emphasize the functional and representational features of the model. The purpose of each program element, its data structure, and its listing are provided.

3. Special Features. The focus of this annex will be on describing the program elements that make up the model. Several features of SEAC, however, deserve to be highlighted. These aspects are either so fundamental, important, or pervasive that they must be understood before understanding the model.

a. SIMULA's CLASS SIMULATION. Computer simulations of real world situations can take several forms: where the system follows well understood physical laws, such as a nuclear power plant, the process can be emulated using mathematical representations (typically sets of differential equations) to study the effects of changes in different componenets; for those systems that are subject to random or unpredicatable occurences computer models have been developed which follow a discrete event approach. Various computer languages have been developed to facilitate modeling using the latter approach. These languages generally are categorized by whether they use event, activity, or process oriented techniques. The language used in SEAC, SIMULA, is a general purpose language, that happens to be very adaptable to simulation modeling. One of the system CLASSES provided within SIMULA is SIMULATION. When prefixed to a CLASS, SIMULATION makes available the necessary software tools to perform discrete event modeling using the "process view of simulation." This approach

involves the decomposition of the system under consideration into the processes that control actions within the target system.

i. Processes. SIMULATION contains a CLASS, called PROCESS, that provides the synchronization mechanism to allow concurrent operation of individual objects. Within SEAC such objects are used to represent the dynamic aspects of the model: the arrival of troops, engineer unit changes or project insertion, and periodic damage done by enemy ground and air actions. The PROCESSES in SEAC essentially control what happens. Those actions taken together provide the scenario, or script, to which SEAC reacts.

ii. Lists. In addition to processes, SIMULATION also provides list processing capability. Because of the dynamic nature of systems being modeled, things frequently come and go within a simulation. To save such elements in an array structure may be either inefficient, if much of the array goes unused, or restrictive, if a particular invocation of the model generates more elements than the array can contain. For such situations SEAC instead relies on SIMULATION CLASSES LINK and HEAD to implement two-way lists. These lists are sets of objects in which, in addition to the element data, each member of the set identifies its predecessor and successor, if they exist, in the list. The set itself is associated with the HEAD object, whose successor and predecessor are the first and last members in the list. Lists are especially convenient when locating list different members is not often needed. For example a list would be most appropriate to implement a first-in-first-out or last-in-first-out processing discipline. In effect the list is a stack which can be viewed from both directions. SEAC employs lists extensively, and in so doing greatly reduces reliance on dimensioned arrays.

b. Model. SEAC is a large and complicated program, having over 5000 lines of code. SEAC is not, however, measurably large in the same way that FORTRAN programs are large, i.e. lines of code and amount of memory required. Fixing SEAC's memory requirements is difficult; it operates on a machine having only 3 megabytes of main memory, but utilizes a virtual memory operating system. Actual memory needs are largely problem specific. First, since memory is dynamically allocated (using heaps for CLASS objects), SEAC's memory usage expands and contracts based on the model's needs. This dynamic memory allocation obviates the need for the programmer/analyst to make data size specifications usually associated with dimensioning arrays. On the other hand dynamic allocation makes it easy to ignore memory requirements during system design only to find out during program testing that there is a problem and difficult restructuring changes are necessary. The size of SIMULA programs in terms of lines of code is also elusive. A subCLASS implicitly contains all the declarations and operations of its parent plus any other instructions and declarations that may be defined in its own body. Also, SIMULA's free format allows multiple statements on one line, and its dot notation (remote accessing) is an extremely compact, as well as powerful, way to reference external data and procedures.

i. AVL Trees. AVL (Adel'son-Vel'skii Landis) Trees (see any standard text on information structures such as Volume 1 of Knuth's series) are used throughout SEAC. They are a means of structuring the various reference and operational data into balanced search trees. With frequent references to factors, catcodes, components, etc., it was necessary for SEAC to be able to process such queries efficiently. Two-way lists are not efficient if the set

is large and many random accesses are anticipated, since searches are necessarily sequential. AVL trees are an alternative in this situation. If the data frequently changes (i.e. additions and deletions), however, the administrative burden of re-balancing the tree structure offsets the gains associated with reduced search time. The power of SIMULA is probably no better evidenced than in the ease of implementing AVL trees in SEAC. Two CLASSES, ROOT and NODE (analogous to HEAD and LINK), define the mechanism. By prefixing them to the subCLASSES representing the various data and operational attributes of the entries, the trees are implemented and the mechanics of how the tree is rebalanced after insertions is transparent to the user.

ii. Modularization. Modularization in SEAC can refer to two different but related purposes: modules to be separately compiled, and modules that provide useful data and procedures (e.g., CLASS SIMULATION can be thought of as a module in that sense). The tiein is that in SIMULA both of these purposes are achieved using the ubiquitous CLASS. This paragraph addresses the separate compilation feature; the section that follows shows the CLASS concatenation structure. Because of the amount of work that the SIMULA compiler must do, compilations are slow. In the early stages of development, the entire model could be comfortably contained in a single program unit. As the code grew, and the amount of settled code grew in relation to developing code, compile times increased measurably, especially when there were other users on the system (wall time in particular, since cpu time is only marginally effected by user loads). SIMULA compilations require that a program unit be compiled after any referenced CLASSES and before any units that use it. What this means is that each time a CLASS is changed and recompiled, all the CLASS or program units that use that CLASS must be recompiled. In forming modules, to break SEAC into more manageable units/modules, consideration was given to segregating CLASSES that were unlikely to be changed as high in the module hierarchy as possible to reduce unnecessary recompilations. Where practical, CLASSES and subCLASSES were grouped together if part of a CLASS/subCLASS hierarchy. While the primary contents of each module are CLASS and PROCEDURE, there are a few global variables defined within each of the modules which because of scoping rules are available at the level they are declared, and any lower levels. The modules or program units used in SEAC are shown in Figure B-1. Everything defined in the various external CLASSES is available within the block of code that controls execution of the simulation.

SEAC Program Modules

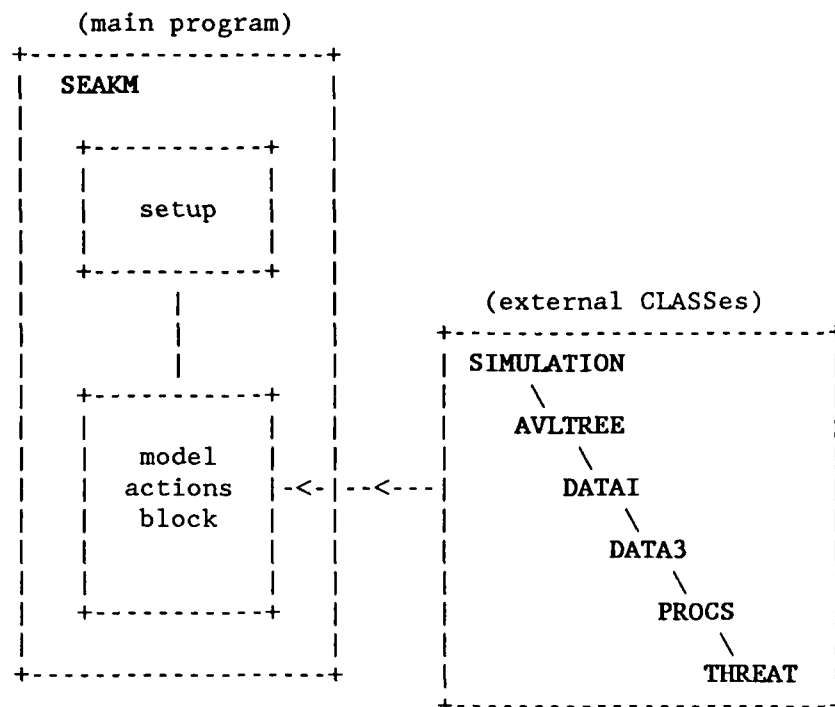


Figure B-1

The external CLASSES are defined in a hierarchical sense--as opposed to being conceptually separate entities. Thus the main program BEGIN...END block prefixed with THREAT has all the data structures and procedures of SIMULATION, AVLTREE, etc. available to it because THREAT is a subCLASS of PROCS, which in turn is a subCLASS of DATA3, etc. In the module description paragraph that follows, the type and purpose of the variables at the outer level of each module or program in SEAC are defined. Note that in terms of scope-visibility rules, declared variables are accessible in the CLASS module they are defined and all other CLASS modules that have the defining CLASS in its prefix structure.

iii. CLASS Hierarchy. The stress on top-down construction is rather pervasive in a model written in SIMULA. The concatenation of CLASSES and the CLASS prefixing necessary for separate compilation of program elements enable the system designer to "think" hierarchy. In so doing the programmer/modeler can create his own data and functional abstractions that map the problem domain into the modeling domain. Figure B-2 shows the CLASSES that are defined in SEAC. It is constructed to show the subCLASS structure of each CLASS. For example, BUILDING is a subCLASS of FACILITY, which in turn is a subCLASS of LINK. While one can gain no insight in how things are done within SEAC, this chart does give one an idea of the types of things that are included in the model. (This chart is also the order that CLASSES are described in this annex.)

Class Hierarchy

COMPONENT			
ENGRPOOL			
FACELEM			
MAPCELL			
NKBASES			
POLICY			
REPORTER	REPORTS		
ROOT	CATREE		
	FACTREE		
	ECAPTREE		
	UOMTREE		
	GEOTREE		
	REALTYTREE		
TGTPREF			
HEAD	ORDLIST		
	COMPO		
	ENGRUNIT		
		SURFACE	RUNWAY
			PAVEMENT
			HARDSTAND
			COMO
			OPNS
			REVETMENT
		BUILDING	PIER
			SHOP
			STORAGE
			MEDICAL
			ADMIN
			QTRS
			POLFACIL
		POL	TANK
			PIPE
			POWER
		UTILITY	WASTE
			WATER
		TRANSP	ROAD
			RR
LINK	FACILITY		

LINK	NODE	CATCOD	
		FACTOR	FACTORX
		ENGRCAP	
		UNOFMS	
		GEOLOC	GEOSUB
	ORDNANCE	PROPERTY	
		BOMB	
		ROCKET	
		DEMOL	
	AUNIT REGCELL REGION HITS TASK		
	INSTALLATION	AIRBASE	AIRPOD
		CAMP	CENTER
		PORT	SEAPOD
		ADA	
		HOSPITAL	
	AIRCRAFT	DEPOT	
		ELEC	
	MUNITION PLANE NKAIRBASE SPFUNIT TEAMORG	BMBR	
		FTR	
	PROCESS	RSD	
		CHANGE	
		MISSION	
		MISGEN	
		REARAXN	

Figure B-2 (continued)

4. Class Description Format. SEAC employs a software design approach usually referred to as object-oriented programming. Objects are in fact an instance of the more general software construct called a CLASS (see Annex A for a discussion of the programming environment). One of the benefits of object-oriented programming is that the code more closely follows the real world system that is being represented or, in our case, modeled. With much of the structure and scope of SEAC bound to these objects, the model can be visualized by looking at the CLASSES that comprise it. The descriptions that follow will employ a standard format: first the purpose of the CLASS will be briefly described; second the attributes and parameters of the CLASS will be outlined; and finally the CLASS itself will be listed. Figure B-3 shows the format of the second portion of each CLASS description. The table is divided into several sections: CLASS name, CLASS prefix, parameter specifications and descriptions (including VIRTUAL designations), variable specifications and descriptions, and finally procedure specifications and descriptions. Other than the CLASS name, all entries in the skeleton are optional, so many CLASSES will not have entries in every portion of the table. Between looking at the purpose and scanning the table, a reader can get a good idea of what the model is all about without getting into the actual code. For those wanting a fuller understanding of the structure, i.e. the code, SIMULA must be studied (See Annex A), and especially the concatenation of CLASSES through CLASS prefixes to understand inheritance and virtual concepts. (Note that the description format is also followed (with slight variations) for descriptions of the major modules that comprise SEAC and for the various procedures and functions that were also defined for the model.)

General Format of Class Variable/Parameter Tables

CLASS name	CLASS prefix (if any)	
CLASS parameter specifications		
<parameter> <type>* <description>		
& virtual declarations		
variable attribute specifications		

procedural attribute definitions		

*/ several conventions will be used for designating "types":
value types for integer, real, character, and boolean
types will be in lower case
object references will be in upper case to highlight
such variables
text references will be in lower case, although they
are actually system objects
arrays will be indicated by the brackets "[]" but will
not identify index ranges, since in many cases that
is not determined until runtime

Figure B-3

5. Module Descriptions. SIMULA supports the development of modular programs. The following general descriptions give an overview of the module hierarchy used in SEAC. AVL TREE is in reality a general purpose library module that provides the software constructs to use balanced trees within the model. The other modules group CLASSES together according to function and scope. The subCLASSES of FACILITY are all found in module DATA2; and PROCESS CLASSES are not declared until module PROCS because of their need to reference installation, facility, catcode, engineer units, etc. data all of which must be either previously or coincidentally defined. (Note that in the following listings the CLASSES that are present in each module are listed without specifications and bodies, since individual descriptions appear later in this annex.)

a. Module AVL TREE.

i. General Contents. This CLASS module contains the NODE and ROOT CLASS definitions necessary to implement the Adel'son-Vel'skii Landis (AVL) balanced tree information structure that is used extensively throughout SEAC. The actual operative code for these CLASSES appear later under their own titles.

ii. Module/CLASS Declarations.

Declarations and Specifications for Module AVLTREE

```
SIMULATION CLASS AVLTREE;  
  BEGIN  
    LINK CLASS NODE.....;  
    CLASS ROOT.....;
```

Figure B-4

iii. Variable Definitions.

Variable Parameters and Attributes for Module AVLTREE

MODULE: AVLTREE	PREFIX: SIMULATION
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	

Figure B-5

b. Module DATAI.

i. General Contents. This module defines many of the element CLASSes used in SEAC. These include those related to engineer units, category codes, planning factors, and enemy weapon systems.

ii. Module/CLASS Declarations.

Top-level Declarations and Specifications for Module DATAI

```
AVLTREE CLASS DATAI(L_F,E_F,R_F,NUMCOUNTRIES,
                    NUMCELLS,VALIDCOUNTRIES,VALIDSERVICES,ATKLOG);
REF(PRINTFILE) L_F,E_F,R_F;
BOOLEAN ATKLOG;
TEXT VALIDCOUNTRIES,VALIDSERVICES;
INTEGER NUMCOUNTRIES,NUMCELLS;
BEGIN
REF(CATREE)   CATAB;
REF(FACTREE)  FACTAB;
REF(ECAPTREE) ECAPTAB;
REF(UOMTREE)  UOMTAB;
REAL ARRAY ENGBLDUP(1:NUMCOUNTRIES,0:9),ENGRAVL(1:NUMCOUNTRIES);
REAL PI,R180,R360;
INTEGER UDAM,UDUD,XUS,XARMY;
CHARACTER XISTD,XTSTD;

NODE CLASS CATCOD.....;
LINK CLASS COMPO.....;
CLASS COMPONENT.....;
ROOT CLASS CATREE.....;
NODE CLASS FACTOR.....;
FACTOR CLASS FACTORX.....;
CLASS FACELEM.....;
ROOT CLASS FACTREE.....;
CLASS ENGRPOOL.....;
LINK CLASS ENGRUNIT.....;
ROOT CLASS ECAPTREE.....;
NODE CLASS ENGRCAP.....;
ROOT CLASS UOMTREE.....;
NODE CLASS UNOFMS.....;
LINK CLASS ORDNANCE.....;
ORDNANCE CLASS BOMB.....;
ORDNANCE CLASS ROCKET.....;
ORDNANCE CLASS DEMOL.....;
HEAD CLASS ORDLIST.....;
```

Figure B-6

iii. Variable Definitions.

Variable Parameters and Attributes for Module DATAI

MODULE: DATAI		PREFIX: AVLTREE
PARAMETERS:		
L_F	PRINTFILE	log file object
E_F	PRINTFILE	error file object (future)
R_F	PRINTFILE	report file object (future)
NUMCOUNTRIES	integer	number of nations in this scenario
VARIABLE ATTRIBUTES:		
CATAB	CATREE	catgory code table
FACTAB	FACTREE	planning factor table
ECAPTAB	ECAPTREE	engineer unit capability table
UOMTAB	UOMTREE	unit of measure table
ENGRBLDUP	real[]	strength buildup in new engr units
ENGRAVL	real[]	general availability rates
PI	real	the constant pi = 22/7
R180	real	radians in 180 degrees
R360	real	radians in 360 degrees
UDAM	integer	random number seed
UDUD	integer	random number seed
XUS	integer	constant for US id
XARMY	integer	constant for Army
XISTD	integer	constant for initial standard
XTSTD	integer	constant for temporary standard

Figure B-7

c. Module DATA2.

i. General Contents. This module defines many of the major SEAC CLASSES. REGIONS, MAPCELLs, GEOLOCs, INSTALLATIONS, FACILITYs, and TASKs are the major elements that are defined in this external CLASS.

ii. Module/CLASS Declarations.

Top-level Declarations and Specifications for Module DATA2

```
DATA1 CLASS DATA2(XINSTAL,XFACS,XDAYS,PIECEWORK,WORKLOG);
  INTEGER XINSTAL,XFACS;
  REAL XDAYS,PIECEWORK;
  BOOLEAN WORKLOG;
  BEGIN
    REF(POLICY) ARRAY CNSTRPLCY,RSTRPLCY(1:XINSTAL,1:NUMCOUNTRIES),
      DAMPLCY,MNTNPLCY(1:XFACS,1:NUMCOUNTRIES);
    REF(REGION) ARRAY RGNSTRUC(1:NUMCOUNTRIES);
    REF(MAPCELL) ARRAY AREAS(1:NUMCELLS);
    REF(GEOTREE)GEOTAB;
    REF(TGTPREF) ARRAY TGTDEFAULTS(1:XINSTAL);
    TEXT FACLIS,CASES,INSTLIS;
    REAL ARRAY CUMRQTS(1:NUMCOUNTRIES,1:4,0:XDAYS,1:3);
    REAL ARRAY SUPRFICL,STRUCT(1:XFACS,1:2), CRTSPLIT(1:3),
      EMDTAB(1:XFACS), SPNLEN,DRPSPAN(1:2);
    REAL BRGFAC,RRDAMFAC,RDDAMFAC,TNKDAMFAC,HTANK,
      HBLAD,PIPDAMFAC,STAGETIME;
    INTEGER PICKSEED,UTSEL,XPRIO,UHOLE;
    CHARACTER XFRAC,XWHL;
```

```
LINK CLASS AUNIT.....;
NODE CLASS GEOLOC.....;
GEOLOC CLASS GEOSUB.....;
ROOT CLASS GEOTREE.....;
NODE CLASS PROPERTY.....;
ROOT CLASS REALTYTREE.....;
LINK CLASS REGCELL.....;
CLASS MAPCELL.....;
LINK CLASS REGION.....;
LINK CLASS HITS.....;
LINK CLASS TASK.....;
LINK CLASS FACILITY.....;
FACILITY CLASS SURFACE.....;
FACILITY CLASS BUILDING.....;
FACILITY CLASS PETRO.....;
FACILITY CLASS UTILITY.....;
FACILITY CLASS TRANSP.....;
SURFACE CLASS RUNWAY.....;
SURFACE CLASS PAVEMENT.....;
PETRO CLASS POLFACIL.....;
PETRO CLASS TANK.....;
PETRO CLASS PIPE.....;
BUILDING CLASS COMO.....;
BUILDING CLASS OPNS.....;
BUILDING CLASS REVETMENT.....;
BUILDING CLASS PIER.....;
BUILDING CLASS SHOP.....;
BUILDING CLASS STORAGE.....;
BUILDING CLASS MEDICAL.....;
BUILDING CLASS ADMIN.....;
BUILDING CLASS QTRS.....;
UTILITY CLASS POWER.....;
```

```

UTILITY CLASS WASTE.....
UTILITY CLASS WATER.....
TRANSP0 CLASS ROAD.....
SURFACE CLASS HARDSTAND.....
TRANSP0 CLASS RR.....
LINK CLASS INSTALLATION.....
CLASS TGTPREF.....
CLASS POLICY.....
CLASS REPORTER.....

```

Figure B-8

iii. Variable Definitions.

Variable Parameters and Attributes for Module DATAI

MODULE: DATA2		PREFIX: DATAI
PARAMETERS:		
XINSTAL	integer	number of INSTALLATION subCLASseS
XFACS	integer	number of FACILITY subCLASseS
XDAYS	integer	scenario length
PIECEWORK	real	threshold for partial day work
WORKLOG	boolean	flag to force work reporting
VARIABLE ATTRIBUTES:		
CNSTRPLCY	POLICY[]	construction policies for install.
RSTRPLCY	POLICY[]	future use for restoration policies
DAMPLCY	POLICY[]	repair policies for facilities
MNTNPLCY	POLICY[]	future use for maintenance policies
RGNSTRUC	REGION[]	point to top country region
AREAS	MAPCELL[]	array of MAPCELL references by id#
GEOTAB	GEOTREE	AVL tree of GEOLOC entries
TGTDEFAULTS	TGTPREF[]	facility target allocation by instl
FAC LIS	text	list of FACILITY SUBCLASS NAMES
CASES	text	list of work categories (eg., const)
INSTLIST	text	list of INSTALLATION CLASS names
CUMRQTS	real[]	cumulative requiremnts (time, etc.)
SUPRFICL	real[]	FACILITY superficial dmg. threshold
STRUCT	real[]	FACILITY structural damage threshld
CRTRSPLIT	real[]	crater repair priority distribution
EMDTAB	real[]	FACILITY effective miss distances
SPNLEN	real[]	notional span lengths for bridges
DRPSPAN	real[]	amount of explosive to drop a span
BRGFAC	real	adjustment to bridge eff miss dist
RRDAMFAC	real	amount of track damaged/lb explosiv
RDDAMFAC	real	amount of road damaged/lb explosive
TNKDAMFAC	real	amount explosive necessary for kill
HTANK	real	nominal height of storage tank
HBLAD	real	nominal height of bladder tank
PIPDAMFAC	real	length pipe damaged/ lbs explosive
STAGETIME	real	not used **

PICKSEED	integer	a random number seed
UTSEL	integer	a random number seed
XPRI0	integer	max priority value
UHOLE	integer	a random number seed
XFRAC	character	character indicating "part" - 'F'
XWHL	character	character indicating "whole" - 'W'

Figure B-9

d. Module PROCS.

i. General Contents. This module defines those procedures and processes that control SEAC's operations. Several procedures (BUILDMAP and ADDASSET) are invoked at startup time to set up the theater and installations. Processes RSD and CHANGE are also invoked at the beginning, but they will remain active as long as units continue to arrive in theater and engineer unit changes or projects are indicated on the orders file.

ii. Module/CLASS Declarations.

Top-level Declarations and Specifications for Module PROCS

```
DATA2 CLASS PROCS(COMBATIDS,RECEP TIME,STAGETIME);
  REAL RECEP TIME,STAGETIME;
  TEXT COMBATIDS;
  BEGIN

  INSTALLATION CLASS AIRBASE.....;
  AIRBASE CLASS AIRPOD.....;
  INSTALLATION CLASS CAMP.....;
  CAMP CLASS CENTER.....;
  INSTALLATION CLASS PORT.....;
  PORT CLASS SEAPOD.....;
  INSTALLATION CLASS ADA.....;
  INSTALLATION CLASS HOSPITAL.....;
  INSTALLATION CLASS DEPOT.....;
  INSTALLATION CLASS ELEC.....;
  PROCESS CLASS RSD.....;
  PROCESS CLASS CHANGE.....;
```

Figure B-10

iii. Variable Definitions.

Variable Parameters and Attributes for Module PROCS

MODULE: PROCS		PREFIX: DATA2
PARAMETERS:		
COMBATIDS	text	characters for which DEPLOY is true
RECEP TIME	real	time unit spends in reception ctr
STAGETIME	real	time unit spends in staging center
VARIABLE ATTRIBUTES: none		

Figure B-11

e. Module THREAT.

i. General Contents. This module provides the program elements that are used to define threat capability and carry out designated attacks against COMMZ facilities.

ii. Module/CLASS Declarations.

Top-level Declarations and Specifications for Module Threat

```

PROCS CLASS THREAT;
BEGIN
  INTEGER UKILL,USUC,UBEFOR,UDAM,UHIT,UREP,URDY,UHIT;
  REAL RELRATE,LBSTGT,PKILL,PSUC,PBEFOR,PDAM,PHIT,XREP;
  REAL ARRAY FACHIT(1:9);
  REF(NKBASES)NKAB;
  REF(HEAD) PLANETYPES;
  REF(ORDLIST)ORDTAB;

  CLASS NKBASES.....;
  LINK CLASS AIRCRAFT.....;
  AIRCRAFT CLASS BMBR.....;
  AIRCRAFT CLASS FTR.....;
  LINK CLASS MUNITION.....;
  LINK CLASS PLANE .....;
  PROCESS CLASS MISSION.....;
  LINK CLASS NKAIRBASE.....;
  PROCESS CLASS MISGEN.....;
  LINK CLASS SPFUNIT.....;
  LINK CLASS TEAMORG.....;
  PROCESS CLASS REARAXN.....;

```

Figure B-12

iii. Variable Definitions.

Variable Parameters and Attributes for Module THREAT

MODULE: THREAT		PREFIX: PROCS	
PARAMETERS: none			
=====			
VARIABLE ATTRIBUTES:			
UKILL	integer	random number seed	
USUC	integer	random number seed	
UBEFOR	integer	random number seed	
UDAM	integer	random number seed	
UHIT	integer	random number seed	
UREP	integer	random number seed	
URDY	integer	random number seed	
UFHIT	integer	random number seed	
RELRATE	real	bombs released per pass	
LBSTGT	real	lbs of explosive per target	
PKILL	real	P(plane destroyed hit)	
PSUC	real	P(mission success plane at target)	
PBEFOR	real	P(plane hit before reaching tgt)	
PDAM	real	not used **	
PHIT	real	P(plane hit)	
XREP	real	repair time parameter	
FACHIT	real[]	Ps(hit/facility)	
NKAB	NKBASES	object containing enemy bases	
PLANETYPES	HEAD	list of plane types	
ORDTAB	ORDLIST	list of ordnance types	
=====			

Figure B-13

f. Module SEAKM.

i. General Contents. This is the main program for SEAC. Its principal role is to read in the file directory that will be used for the scenario, initialize various scenario parameters, and initiate the operation of the model. The only exception is the definition of reports within the REPORTER CLASS. This is done to provide the user easier access to output routines.

ii. Module/CLASS Declarations.

Top-level Declarations and Specifications for SEAKM

```

REF(INFILE)RUNCNTRL;
REF(PRINTFILE)LOGFYL,ERRFYL,REPFYL;
TEXT COUNTRIES,SERVICES,RSDFLAGS,TITLE;
TEXT F_MISSIONS,F_REGIONS,F_CATCODES,F_FACTORS,F_ORDERS,F_RNGRCOM,F_UOM;
TEXT F_ARRIVALS,F_ASSETS,F_GEOLOC,F_CAPABILITY,F_POLICY,F_AIRTHREAT;
TEXT IODIRECTIONS,F_LOG,F_ERR,F_REP;
INTEGER NXCELLS,NXCOUNTRIES,NXSERVICES,NXINSTALLATIONS,NXFACILITIES,NRPTS;
BOOLEAN ECHO,DUMP,REPCAP,DMPWRK,DMPATK;
REAL SIMLENGTH,RCPTIME,STGTIME,DOTASK;
REF(OUTFILE) MONITOR;
REAL ELAPSED,DELTIME;
INTEGER CPUT;

```

Figure B-14

iii. Variable Definitions.

Variable Parameters and Attributes for Module SEAKM

MODULE: SEAKM		PREFIX: none
PARAMETERS: not applicable (main program, not an external CLASS)		
VARIABLE ATTRIBUTES:		
RUNCNTRL	INFILE	file object controlling execution
LOGFYL	PRINTFILE	file for all output messages
ERRFYL	PRINTFILE	error file (future)
REPFYL	PRINTFILE	report file (future)
COUNTRIES	text	valid country characters
SERVICES	text	valid service characters
RSDFLAGS	text	valid deployment flags (FRQN)
TITLE	text	title to associate with run
F_MISSIONS	text	name of air mission file
F_REGIONS	text	name of region-mapcell file
F_CATCODES	text	name of catcode-component file
F_FACTORS	text	name of planning factors file
F_ORDERS	text	name of change file (exog events)
F_RNGRCOM	text	name of ground threat file
F_UOM	text	name of unit of measure equiv file
F_ARRIVALS	text	name of TROOP (unit arrivals) file
F_ASSETS	text	name of facility asset file
F_GEOLOC	text	name of GEOLOC-installation file
F_CAPABILITY	text	name of engineer unit capab file
F_POLICY	text	name of priority-policy file
F_AIRTHREAT	text	name of air threat aob+ file
F_LOG	text	name of log output file
F_ERR	text	name of error report file
F_REP	text	name of results report file
IODIRECTIONS	text	buffer containing f_* names
NXCELLS	integer	maximum number of mapcells

NXCOUNTRIES	integer	number of countries in scenario
NXSERVICES	integer	number of services in scenario
NXINSTALLATIONS	integer	number of installation subCLASSES
NXFACILITIES	integer	number of facility subCLASSES
NRPTS	integer	number of report regions
ECHO	boolean	indicates input data printing
DUMP	boolean	indicates data table printing
REPCAP	boolean	indicates daily summary of capabltiy
DMPWRK	boolean	indicates capability-task printing
SIMLENGTH	real	scenario duration
RCPTIME	real	time unit spends in reception ctr
STGTIME	real	time unit spends in staging base
DOTASK	real	threshold percent for partial work
ELAPSED	real	total elapsed cpu time
DELTIME	real	cpu time used during last simul-day
CPUT	integer	total elapsed cpu time (c\$tima)
MONITOR	OUTFILE	file object for progress messages

Figure B-15

6. Class Descriptions. The following summary descriptions and program listings are a complete definition of SEAC's CLASS environment. The order of the sections corresponds to the CLASS-subCLASS Hierarchy order in figure B-2.

a. Class COMPONENT.

i. Purpose. Objects from class COMPONENT contain detailed engineer manhour information for either the construction or repairing of facilities of a given size. Components reflect manhour data defined for select facilities from the military services' functional component systems. (This is also an example of a CLASS that is only a data structure, i.e. there are no operations or procedures associated with it.)

ii. Class Skeleton.

Parameters and Attributes for Class COMPONENT

CLASS: COMPONENT		PREFIX: none
PARAMETERS:		
UOM	text	unit of measure for component
PART	character	fractionalization indicator
SIZE	real	size of defined facility component
HORZ	real	daily horizontal manhours to perform
VERT	real	daily vertical manhours to perform
OTH	real	daily other hours to perform
MINDAY	integer	minimum number of days to complete
PROCEDURE ATTRIBUTES: none		

VARIABLE ATTRIBUTES: none		

Figure B-16

iii. Listing.

Listing of Class COMPONENT

```
CLASS COMPONENT(UOM,PART,SIZE,HORZ,VERT,OTH,MINDAY);  
  REF(UNOFMS)UOM;  
  CHARACTER PART; INTEGER MINDAY;  
  REAL HORZ,VERT,OTH,SIZE; ;
```

Figure B-17

b. Class ENGRPOOL.

i. Purpose. Engineer units are explicitly identified and manipulated in SEAC (see ENGRUNIT). Units are assigned to either an installation or a region. Since units may be moved into or out of either type of location each day, engineer capability at those locations must be calculated on a daily basis. Reporting requirements also required tallying engineer capability and utilization by service and priority respectively. Since many (most?) installations and regions do not have engineer units assigned it was deemed undesirable to have many empty arrays competing for scarce computer memory. The ENGRPOOL was therefore defined to save storage as well as provide the structure to monitor engineer capability and use. Objects from this CLASS are attached only to locations that have engineer units.

ii. Class Skeleton.

Parameters and Attributes for Class ENGRPOOL

CLASS: ENGRPOOL		PREFIX: none	
PARAMETERS: none			
VARIABLE ATTRIBUTES:			
HC	real	total horizontal capability available	
		that day in manhours	
VC	real	as above but for vertical capability	
OC	real	as above but for general capability	
ENGRUSE	real[]	records daily utilization of capability	
		by skill and priority	
ESVC	real[]	records daily capability by service	

PROCEDURE ATTRIBUTES:			
TALLY		called each day to calculate the total	
		available manhours at the installation	
		or region to which the the CLASS	
		belongs	

Figure B-18

iii. Listing.

Listing of Class ENGRPOOL

```

CLASS ENGRPOOL;
  BEGIN
    REAL HC,VC,OC;REAL ARRAY ENGRUSE(1:4,1:3),ESVC(1:5);

    PROCEDURE TALLY(ES,RP); REF(HEAD)ES;BOOLEAN RP;
      BEGIN
        REF(ENGRUNIT) E;INTEGER I;
        FOR I := 1,2,3,4 DO
          ENGRUSE(I,1):=ENGRUSE(I,2):=ENGRUSE(I,3):=0.0;
        FOR I := 1,2,3,4,5 DO ESVC(I) := 0.0;
        HC := VC := OC := 0.0;
        E :- ES.FIRST;
        WHILE E /= NONE DO
          BEGIN
            E.MUSTER(HC,VC,OC,RP,ESVC(E.UTCMSTR.SERVICE));
            E :- E.SUC;
          END;
        END -- TALLY --;
      END ** ENGRPOOL **;

```

Figure B-19

c. Class FACELEM.

i. Purpose. Much of the facility requirements determined in the model result from the use of factors. Factors (see CLASS FACTOR) are associated with units, equipment items, installation types, etc. Each factor contains a list of FACELEM objects which define either the amount and type of required facilities, or other factors that are triggered by that factor.

ii. Class Skeleton.

Parameters and Attributes for Class FACELEM.

CLASS: FACELEM		PREFIX: none
PARAMETERS:		
CODE	text	either the identity of the required facility (JCS catcode) or other factor to be evaluated (usually an equipment item) for the owning factor
UOM	text	the unit of measure for a catcode
RATE	real	the amount per factor unit to construct
VARIABLE ATTRIBUTES:		
NX	FACELEM	points to the next object in the factor list
PROCEDURE ATTRIBUTES: none		

Figure B-20

iii. Listing.

Listing of Class FACELEM.

```
CLASS FACELEM(CODE,RATE,UOM);
  VALUE CODE,UOM;TEXT CODE,UOM;REAL RATE;
  BEGIN REF(FACELEM) NX; END ** FACELEM **;
```

Figure B-21

d. Class MAPCELL.

i. Purpose. MAPCELL objects partition the theater of operations into geographical areas. Within each MAPCELL are the MSR assets that exist or are built, a list of the GEOLOCs that fall within its boundary (which in turn contain the installations identified by that GEOLOC), and pointers to the regions that contain the MAPCELL. The hierarchical structure by which SEAC portrays a theater was created to specifically to address the possibility of overlapping engineer unit responsibility. (It was also to improve upon the rather restrictive Base Complexing scheme implemented in CESPg which tended to treat grouped GEOLOCs as a single isolated installation.) SEAC uses the combination of REGIONS (for better representation of engineer capability in area support roles), MAPCELLS (to represent an actual physical portion of the theater and indicate what MSR and GEOLOCs are found in it), and INSTALLATIONS (to represent different facility assets and requirements at those locations

where units are assigned). Presently three types of MSRs are played: roads, railroads and pipelines. SEAC also requires that the nation responsible for MSR construction or repair be indicated in each MAPCELL.

ii. Class Skeleton.

Parameters and Attributes for Class MAPCELL

CLASS: MAPCELL		PREFIX: none
PARAMETERS:		
RRESP	integer	identifies nation responsible for roads
RRESP	integer	as above, but for railroads
PIPRESP	integer	as above, but for pipeline facility
VARIABLE ATTRIBUTES:		
GEOSET	HEAD	set of geolocs in the mapcell
ROADS	HEAD	set of road facilities in mapcell
RRS	HEAD	set of railroad facilities
PIPELINE	HEAD	set of pipeline facilities
TASKLIST	HEAD	set of outstanding engineer work tasks for MSR construction and repair in the mapcell
DMGLIST	HEAD	list of new war damage for MSR facilities occurring during day
RGNLINK	REGION[]	vector mapping the mapcell into each nations regional heirarchy
TASKPTR	TASK	points to work task being considered for accomplishment that day
AOR	boolean	true if mapcell is in COMMZ
INDX	integer	index of the mapcell
PROCEDURE ATTRIBUTES:		
NEEDS		invokes requirement generation for the mapcell and all installations found within it
DMGCNTRL		evaluates the facility damage data found on the DMGLIST
HASA	boolean	inserts MSR assets into the mapcell or schedules a new MSR project to construct a facility

Figure B-22

iii. Listing.

Listing of Class MAPCELL

```
CLASS MAPCELL(RDRESP,RRESP,PIPRESP);
  INTEGER RDRESP,RRESP,PIPRESP;
  BEGIN
    REF(HEAD) GEOSSET,ROADS,RRS,PIPELINE,TASKLIST,DMGLIST;
    REF(REGION) ARRAY RGNLINK(1:NUMCOUNTRIES);
    REF(TASK) TASKPTR; BOOLEAN AOR; INTEGER INDX;

    PROCEDURE NEEDS;
      BEGIN
        REF(GEOLOC)G;REF(INSTALLATION)I;
        IF AOR THEN DMGCNTRL;
        G:-GEOSSET.FIRST;
        WHILE G=/=NONE DO
          BEGIN
            I:-G.SITES.FIRST;
            WHILE I=/=NONE DO
              BEGIN
                I.CHECKIN;
                IF AOR THEN
                  BEGIN
                    I.DAMCNTRL;
                    I.ASSETS.ASSESS(I);
                  END;
                I :- I.SUC;
              END;
            G :- G.SUC;
          END;
        END ** NEEDS **;

    PROCEDURE DMGCNTRL;
      BEGIN REF(HITS)H;
      FOR H :- DMGLIST.FIRST WHILE H /= NONE DO
        BEGIN
          INSPECT H.FAC
            WHEN PIPE DO REPAIR(H.EXTENT,PIPRESP,XARMY,INDX,TASKLIST)
            WHEN ROAD DO REPAIR(H.EXTENT,RDRESP,XARMY,INDX,TASKLIST)
            WHEN RR DO REPAIR(H.EXTENT,RRESP,XARMY,INDX,TASKLIST);
          H.OUT;
        END;
      END -- DMGCNTRL --;

    BOOLEAN PROCEDURE HASA(CC,CN,SVC,UM,AMT,STND);
      INTEGER CN,SVC;TEXT CC,UM;REAL AMT;CHARACTER STND;
      BEGIN
        REF(CATCOD)CTCD;REF(FACILITY)F;
        REF(COMPO)CMP;REF(COMPONENT)CMPNT;

        CTCD :- CATAB.FIND(CC);
        IF CTCD /= NONE THEN
          BEGIN
```

```

CMP :- CTCD.SELECT(CN,SVC);
IF CMP == NONE AND CN <> XUS THEN
    CMP:-CTCD.SELECT(XUS,SVC);
IF CMP /= NONE THEN
    BEGIN
    CMPNT:-IF STND = XTSTD THEN CMP.TEMP
    ELSE
    IF CMP.INTL /= NONE THEN CMP.INTL ELSE CMP.TEMP;
    IF CMPNT /= NONE THEN
    BEGIN
    IF CMPNT.UOM.UOM = UM THEN
        F:-CATOFAC(NONE,CMP,AMT,CC,STND)
    ELSE
        IF CMPNT.UOM.EQVMS = UM THEN
            F:-CATOFAC(NONE,CMP,
                AMT*CMP.INTL.UOM.COEF,CC,STND);
    IF F == NONE THEN
    BEGIN
    OUTTEXT("...MAPCELL CATCODE /UOM ERROR ");
    OUTTEXT(CC);OUTCHAR('/');OUTTEXT(UM);
    OUTIMAGE;
    END
    ELSE
    BEGIN
    IF STND=XISTD THEN
        F.CONSTR(CN,SVC,INDX,TASKLIST)
        ELSE F.BUILT := TRUE;
    IF F IS PIPE THEN
    BEGIN
    IF PIPELINE==NONE THEN PIPELINE:-NEW HEAD;
    F.INTO(PIPELINE);HASA := TRUE;
    END
    ELSE
    IF F IS ROAD THEN
    BEGIN
    IF ROADS==NONE THEN ROADS:-NEW HEAD;
    F.INTO(ROADS);HASA := TRUE;
    END
    ELSE
    IF F IS RR THEN
    BEGIN
    IF RRS==NONE THEN RRS:-NEW HEAD;
    F.INTO(RRS);HASA := TRUE;
    END;
    END;
    END;
    END;
    END -- HASA --;
GEOSSET :- NEW HEAD;DMGLIST:-NEW HEAD;TASKLIST:-NEW HEAD;
INDX:=1;AOR := TRUE;
END ** MAPCELL **;

```

Figure B-23

e. Class NKBASES.

i. Purpose. If the threat portion of SEAC is invoked, an object of CLASS NKBASES is created to encompass all enemy airbases. It provides a means to access enemy airbases and determine the number of planes that are operational at any time.

ii. Class Skeleton.

Parameters and Attributes for Class NKBASES

CLASS: NKBASES		PREFIX: none
PARAMETERS: none		
VARIABLE ATTRIBUTES:		
FIELDS	HEAD	set of enemy airfield locations
PROCEDURE ATTRIBUTES:		
FIND		searches FIELDS for a designated enemy airfield identifier
AOBPOOL		invoked before every mission order to determine plane availability

Figure B-24

iii. Listing.

Listing of Class NKBASES

```
CLASS NKBASES;
  BEGIN REF(HEAD) FIELDS;
  REF(NKAIRBASE) PROCEDURE FIND(L); TEXT L;
    BEGIN
      REF(NKAIRBASE)X;
      X :- FIELDS.FIRST;
      WHILE X /= NONE DO
        IF L = X.ID THEN
          BEGIN FIND :- X; X :- NONE; END
        ELSE
          X :- X.SUC;
        END -- FIND --;
      PROCEDURE AOBPOOL(READY); REAL READY;
        BEGIN
          REF(NKAIRBASE)X;
          OUTIMAGE;
          OUTTEXT("    POOL    BASE    BOMBERS  FIGHTERS    REPAIR");
          OUTTEXT(" (T=");OUTFIX(TIME,2,6);OUTCHAR(')');
          OUTIMAGE;
          X :- FIELDS.FIRST;
          WHILE X /= NONE DO BEGIN X.PREPARE(READY); X:-X.SUC; END;
          END -- AOBPOOL --;
        FIELDS :- NEW HEAD;
      END ** NKBASES **;
```

Figure B-25

f. Class POLICY.

i. Purpose. One of the important features of SEAC is the ability to prioritize work requirements. In the initial application of the model only construction and damage repair were included (it would be relatively easy to add maintenance and restoration work when suitable data become available for all services). The document discusses the ESC priority system for engineer assessments in general and the one used for EAK-COMMZ in particular. In that study ESC settled on assigning priorities in the following manner: new construction tasks receive a priority based on the type of installation on which the facility will be built; damage repair, however, will be based on the facility category without regard to the installation. Initial reaction to the results of EAK-COMMZ, however, seemed to indicate that perhaps it would be better to base new construction priorities also on the facility type (such a change could be easily accommodated). In addition to the priority, ESC also wanted to represent several other elements of work scheduling or assignment: host nation support, national differences in priorities, build/no-build policies, and permitting dynamic rather than static priority assignments. As the components became more numerous ESC realized it was dealing with a broader issue than just priority, and that while it could define several (many?) multi-dimensional arrays internal to SEAC, the declaration of a work "POLICY" # would nicely bind the related topics together.

ii. Class POLICY Skeleton.

Parameters and Attributes for Class POLICY.

CLASS: POLICY		PREFIX: none
PARAMETERS:		
PRIOGRP	integer	the numeric equivalent to the priority group designation associated with the work item.
SHARE	real	indicates what share of the work will be done by military engineers
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES: none		

Figure B-26

iii. Listing.

Listing of Class POLICY.

```
CLASS POLICY(PRIOGRP,SHARE);  
  INTEGER PRIOGRP;REAL SHARE;  
  BEGIN  
    IF SHARE > 1.0 THEN SHARE := .01 * SHARE;  
  END ** POLICY **;
```

Figure B-27

g. Class REPORTER.

i. Purpose. SEAC is a detailed simulation of COMMZ facility requirements and engineer capability to satisfy those needs. Much data is used, generated and discarded in the course of a simulation. Capturing the data and making it available in post execution (or in the future perhaps intra-execution) reports presented a problem of balance. The more data one wanted to collect the greater the computer memory needed to manage that data. But there are both practical and physical limits to the amount of data that can be retained without bringing the model to a computational halt. To get around this problem and at the same time to provide flexibility, SEAC employs the REPORTER CLASS. One can think of a REPORTER object (or an object with REPORTER in its prefix--to wit REPORTS) as someone who goes to a location and collects information about what is happening in that location. In a sense the object is very much like a real reporter. The location in SEAC that can receive these objects are the REGION. Any REGION or subordinate REGION in the regional heirarchy defined for a study can be designated for reporting. Thus a user is able to selectively report on COMMZ areas of interest and not generate and retain information that is not of interest to him. The information that is collected covers: requirements capability, and utilization. It also records service, work category, and priority sub-classifications.

ii. Class REPORTER Skeleton.

Parameters and Attributes for Class REPORTER.

CLASS: REPORTER		PREFIX: none
PARAMETERS:		
RTITLE	text	user designated identification for the REGION being reported
RPRGN	REGION	points to designated REGION
LENTIM	real	(not used)
RESULTS	VIRTUAL	
ONEOUT	VIRTUAL	called to collect requirements and capability data before WORK called (implemented in REPORTS)
TWOUT	VIRTUAL	called after WORK invoked to report on utilization and carryover rqts (implemented in REPORTS)
VARIABLE ATTRIBUTES:		
RQT	real[]	regional requirements by skill and priority for that day
UTLZ	real[]	utilization of total regional capability by skill and priority
CAP	real[]	regional capability by skill for that day
WRKCAT	real[]	requirements that day by service and work type (constr or repair)
SVCAP	real[]	total manhour capability for that region by service
PROCEDURE ATTRIBUTES:		
ZED		zeroes out collection arrays

Figure B-28

iii. Listing.

Listing of Class REPORTER.

```
CLASS REPORTER(RTITLE,RPRGN,LENTIM);
  VALUE RTITLE;TEXT RTITLE;REF(REGION)RPRGN;REAL LENTIM;
  VIRTUAL : PROCEDURE RESULTS,ONEOUT,TWOUT;
  BEGIN
    REAL ARRAY RQT,UTLZ(1:4,1:3),CAP(1:3), WRKCAT(1:5,1:2), SVCAP(1:5);
    INTEGER P;
    PROCEDURE ZED;
      BEGIN
        CAP(1):=-CAP(2):-CAP(3):=0.0;
        FOR P := 1,2,3,4,5 DO
          WRKCAT(P,1):=WRKCAT(P,2):=SVCAP(P):=0.0;
        FOR P := 1, 2, 3, 4 DO
          BEGIN
            RQT(P,1):=-RQT(P,2):-RQT(P,3):=0.0;
            UTLZ(P,1):=UTLZ(P,2):=UTLZ(P,3):=0.0;
          END;
        END -- ZED --;
      END *** REPORTER ***;
```

Figure B-29

h. Class REPORTS.

i. Purpose. Having defined REPORTER objects to collect data, ESC still needed to format how that data would actually be represented in printed reports. To do this it defined REPORTS as a sub-CLASS of REPORTER. (Part of the reason to conceptually separate collection from output was to circumvent compile waiting times since a change to REPORTS output formats requires only that SEAKM be recompiled, while a change to REPORTER would mean that DATA3, PROCS, THREAT, and SEAKM would all be re-compiled -- which could take hours during normal business hours). REPORTS objects do not maintain tables of information or generate multiple files of output data, instead they distill information collected for a REGION, format it for eventual printing, prefix the record with location-time-report stamp, and finally write the record to a designated output file. When the simulation is over the output file is then sorted by the stamp field, stripped of the stamp, and then printed out in proper order. (This last operation is not part of SEAC but is accomplished using system sort and listing commands that are normally available on any computer system.

ii. Class REPORTS Skeleton.

Parameters and Attributes for Class REPORTS.

CLASS: REPORTS		PREFIX: REPORTER
PARAMETERS: none		
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
ONEOUT	VIRTUAL	called to extract capability and requirements data from a region prior to commencing any actual work that day
TWOUT	VIRTUAL	called after work is done for day to record how capability was utilized and what work remains in the TASK queues

Figure B-30

iii. Listing.

Listing of Class REPORTS.

```

REPORTER CLASS REPORTS;
  BEGIN
    REAL SMCAP,SMUTLZ;

    PROCEDURE TAG(REPCOD,TYM);CHARACTER REPCOD;REAL TYM;
      BEGIN
        R_F.OUTCHAR('&');R_F.OUTINT(RPRGN.RCOUNTRY,2);R_F.OUTCHAR(':');
        R_F.OUTINT(RPRGN.RX,3);R_F.OUTCHAR(':');
        R_F.OUTCHAR(REPCOD);R_F.OUTCHAR(':');
        R_F.OUTFIX(TYM,1,5);R_F.OUTCHAR(':');
        END -- TAG --;

    PROCEDURE PLAB;
      BEGIN R_F.OUTTEXT("    DAY    ");
        R_F.OUTTEXT("          V I T A L          C R I T I C A L ");
        R_F.OUTTEXT("          E S S E N T I A L          N E C E S S A R Y ");
        R_F.OUTIMAGE;
        END;
    PROCEDURE SLAB;
      BEGIN
        R_F.SETPOS(25);
        R_F.OUTTEXT("    HORZ    VERT    OTHER    HORZ    VERT
OTHER");
        R_F.OUTTEXT("    HORZ    VERT    OTHER    HORZ    VERT
OTHER");
        R_F.OUTIMAGE;
        END;

```

PROCEDURE ONEOUT;

```

BEGIN
  TAG('C',TIME);
  R_F.OUTFIX(TIME-.4,0,10);R_F.SETPOS(35);
  R_F.OUTFIX(CAP(1),1,10);
  R_F.OUTFIX(CAP(2),1,10);
  R_F.OUTFIX(CAP(3),1,10);
  SMCAP:=CAP(1)+CAP(2)+CAP(3);R_F.OUTFIX(SMCAP,1,10);
  FOR P:= 1,2,3,4,5 DO R_F.OUTFIX(SVCAP(P),1,10);
  R_F.OUTIMAGE;
  TAG('R',TIME); R_F.OUTFIX(TIME-.4,0,6);R_F.SETPOS(25);
  FOR P:= 1,2,3,4 DO
    BEGIN
      R_F.OUTFIX(RQT(P,1),1,9);
      R_F.OUTFIX(RQT(P,2),1,9);
      R_F.OUTFIX(RQT(P,3),1,9);
    END;
  R_F.OUTIMAGE;
  TAG('T',TIME); R_F.OUTFIX(TIME-.4,0,6);R_F.SETPOS(25);
  FOR P:=1,2,3,4,5 DO
    BEGIN
      R_F.OUTFIX(WRKCAT(P,1),1,10);
      R_F.OUTFIX(WRKCAT(P,2),1,10);
    END;
  R_F.OUTIMAGE;
END;
```

PROCEDURE TWOUT;

```

BEGIN
  SMUTLZ := 0.0;
  TAG('U',TIME);
  R_F.OUTFIX(TIME-.4,0,6);R_F.SETPOS(25);
  FOR P:= 1,2,3,4 DO
    BEGIN
      R_F.OUTFIX(UTLZ(P,1),0,8);
      R_F.OUTFIX(UTLZ(P,2),0,8);
      R_F.OUTFIX(UTLZ(P,3),0,8);
      SMUTLZ:=SMUTLZ+UTLZ(P,1)+UTLZ(P,2)+UTLZ(P,3);
    END;
  R_F.OUTFIX( IF SMCAP>0.0 THEN 100.0*SMUTLZ/SMCAP ELSE 0.0
    ,1,6); R_F.OUTIMAGE;
END;
```

```

TAG('C',.1);
R_F.OUTTEXT("CAPABILITY REPORT FOR ");R_F.OUTTEXT(RTITLE);
R_F.OUTTEXT(" (COUNTRY/REGION =");R_F.OUTINT(RPRGN.RCOUNTRY,2);
R_F.OUTCHAR('/');R_F.OUTINT(RPRGN.RX,2);R_F.OUTCHAR(' ');
R_F.OUTIMAGE;
TAG('C',.2); R_F.OUTTEXT(" DAY "); R_F.SETPOS(35);
R_F.OUTTEXT(" HORZ VERT OTHER TOTAL ");
FOR P:= 1,2,3,4,5 DO
  BEGIN R_F.OUTTEXT(SERVICES.SUB(P,1));R_F.OUTTEXT(" ");END;
R_F.OUTIMAGE;
```

```

TAG('R',.1);
R_F.OUTTEXT("REQUIREMENTS REPORT FOR ");R_F.OUTTEXT(RTITLE);
R_F.OUTTEXT(" (COUNTRY/REGION =");R_F.OUTINT(RPRGN.RCOUNTRY,2);
R_F.OUTCHAR('/');R_F.OUTINT(RPRGN.RX,2);R_F.OUTCHAR(')');
R_F.OUTIMAGE;
TAG('R',.2);PLAB;TAG('R',.3);SLAB;
TAG('U',.1);
R_F.OUTTEXT("UTILIZATION REPORT FOR ");R_F.OUTTEXT(RTITLE);
R_F.OUTTEXT(" (COUNTRY/REGION =");R_F.OUTINT(RPRGN.RCOUNTRY,2);
R_F.OUTCHAR('/');R_F.OUTINT(RPRGN.RX,2);R_F.OUTCHAR(')');
R_F.OUTIMAGE;
TAG('U',.2);
R_F.OUTTEXT("      DAY      ");
R_F.OUTTEXT("      V I T A L      C R I T I C A L ");
R_F.OUTTEXT("      E S S E N T I A L      N E C E S S A R Y ");
R_F.OUTTEXT("      TOT");R_F.OUTIMAGE;
TAG('U',.3);
R_F.SETPOS(25);
R_F.OUTTEXT("      HORZ      VERT      OTHER      HORZ      VERT      OTHER");
R_F.OUTTEXT("      HORZ      VERT      OTHER      HORZ      VERT      OTHER");
R_F.OUTTEXT("      3");R_F.OUTIMAGE;
TAG('T',.1);
R_F.OUTTEXT("WORK SUBMISSION REPORT FOR ");R_F.OUTTEXT(RTITLE);
R_F.OUTTEXT(" (COUNTRY/REGION =");R_F.OUTINT(RPRGN.RCOUNTRY,2);
R_F.OUTCHAR('/');R_F.OUTINT(RPRGN.RX,2);R_F.OUTCHAR(')');
R_F.OUTIMAGE;
TAG('T',.2);R_F.OUTTEXT("      DAY      ");
FOR P:= 1,2,3,4,5 DO
    BEGIN
        R_F.OUTTEXT("-----");
        R_F.OUTTEXT(SERVICES.SUB(P,1));
        R_F.OUTTEXT("-----");
    END;
R_F.OUTIMAGE;
TAG('T',.3);R_F.SETPOS(25);
FOR P := 1,2,3,4,5 DO R_F.OUTTEXT(" NEW CNSTR      WAR DMG");
R_F.OUTIMAGE;
END *** REPORTS ***;

```

Figure B-31

i. Class ROOT.

i. Purpose. SEAC keeps a lot of information at hand to conduct the simulation. Catcodes, components, factors, GEOLOCs, engineer unit capability, etc. are referred to throughout execution of the model. Accessing this information becomes a significant part of the overall execution time. There are many different ways to structure the data to facilitate look-up operations: lists, trees, hashing tables, or simple indexed arrays. One of the better techniques uses a balanced binary tree (see discussion of AVL trees). This device starts with the normal binary tree representation where each node has two nodes that belong to it--one whose key is lower and one whose key is higher. Constructing such trees is usually a matter of taking the newest

object and comparing it with successive nodes as it moves through the tree until it finds its proper place. If one blindly follows this assignment strategy for new nodes, however, one may end up with undesirable results (as would happen if the candidate node entries are already sorted, in which case the tree would be a single branch with each node having one subordinate node). One could get around this artifact by randomizing or preprocessing the input, or one could make the tree construction logic smarter. ESC uses the balanced tree device (where the difference in branch lengths is never greater than 1) that was developed by Adelson-Velskii and Landis. A ROOT CLASS is constructed to be a prefix CLASS for objects that will indicate an AVL tree will be used to structure data. (See CLASS NODE for the other component of the AVL tree implementation.) It is through the ROOT object that the tree is identified, searched, or traversed (printed out in key order).

ii. Class ROOT Skeleton.

Parameters and Attributes for Class ROOT.

CLASS: ROOT		PREFIX:
PARAMETERS:		
TITLE	text	title of the tree
HEADING	text	additional labeling text
FIND	VIRTUAL	searches through the tree for an entry with submitted key
VARIABLE ATTRIBUTES:		
FIRST	NODE	points to first node (root node)
PROCEDURE ATTRIBUTES:		
TRAVERSE		initiates action to move through tree in key order, printing node information along the way

Figure B-32

iii. Listing.

Listing of Class ROOT.

```
CLASS ROOT(TITLE,HEADING);
  VALUE TITLE,HEADING; TEXT TITLE,HEADING;
  VIRTUAL : REF(NODE) PROCEDURE FIND;
  BEGIN
    REF(NODE)FIRST;

    PROCEDURE TRAVERSE;
      BEGIN OUTIMAGE;
      OUTTEXT(TITLE); OUTTEXT("      ");
      OUTTEXT(HEADING); OUTIMAGE;
      IF FIRST=-/NONE THEN FIRST.TRAVERSE
      ELSE
        BEGIN OUTTEXT("      TREE IS EMPTY ");OUTIMAGE;END;
      END -- TRAVERSE --;
    END *** ROOT ***;
```

Figure B-33

j. Class CATREE.

i. Purpose. CATREE is the ROOT CLASS for an AVL tree used to store catcode and associated service component data necessary to estimate facility construction and repair. A procedure, BUILD, has also been included that constructs the catcode tree.

ii. Class CATREE Skeleton.

Parameters and Attributes for Class CATREE.

CLASS: CATREE		PREFIX: ROOT
PARAMETERS: none		
VARIABLE ATTRIBUTES: none		

PROCEDURE ATTRIBUTES:		
FIND	Procedure that searches through the tree structure for an entry that matches the submitted key	
BUILD	Constructs the tree. The source file is read and tree entries are created.	

Figure B-34

iii. Listing.

Listing of Class CATREE.

```
ROOT CLASS CATREE;
BEGIN
  REF(CATCOD) PROCEDURE FIND(T); TEXT T;
  BEGIN
    REF(CATCOD) C; INTEGER I;
    C :- FIRST;
    IF C == NONE THEN
      BEGIN FIND :- NONE;
      OUTTEXT(" CATCODES EMPTY");OUTIMAGE;
      END
    ELSE
      WHILE C /= NONE DO
        IF T < C.CATLAB THEN C :- C.LL
        ELSE
          IF T = C.CATLAB THEN BEGIN FIND :- C;C :- NONE; END
          ELSE
            C :- C.RL;
        END ** FIND **;
      END
    END
  END

  PROCEDURE BUILD(ECHO,CATFILE); TEXT CATFILE; BOOLEAN ECHO;
  BEGIN
    REF(INFILE)CATIN; REF(CATCOD)CT;REF(COMPO)CM;REF(COMPONENT)CP;
    INTEGER CN,CS;
    TEXT LSTCODE,CCODE,CNAT,CMAG,CUM,CPART,CHORIZ,CVERT,COTH,CMIN;
    TEXT CSV,CNAME,CTYP;

    CATIN :- NEW INFILE(CATFILE);
    LSTCODE :- BLANKS(4);
    CATIN.OPEN(BLANKS(80));
    CATIN.INIMAGE;

    CCODE :- CATIN.IMAGE.SUB(68,4);      CNAME :- CATIN.IMAGE.SUB(12,18);
    CNAT :- CATIN.IMAGE.SUB(9,1);        CSVC :- CATIN.IMAGE.SUB(10,1);
    CMAG :- CATIN.IMAGE.SUB(31,6);        CUM :- CATIN.IMAGE.SUB(38,2);
    CPART :- CATIN.IMAGE.SUB(41,1);       CHORIZ :- CATIN.IMAGE.SUB(42,7);
    CVERT :- CATIN.IMAGE.SUB(49,7);       COTH :- CATIN.IMAGE.SUB(56,7);
    CMIN :- CATIN.IMAGE.SUB(64,3);        CTYP :- CATIN.IMAGE.SUB(73,1);

    WHILE NOT CATIN.ENDFILE DO
      BEGIN
        IF ECHO THEN
          BEGIN OUTTEXT(">CTC:");OUTTEXT(CATIN.IMAGE);OUTIMAGE;END;
        IF CCODE <> LSTCODE THEN
          BEGIN
            CT :- FIND(CCODE);
            IF CT == NONE THEN
              BEGIN
                CT :- NEW CATCOD(CCODE,CNAME);
                CT.RANK(THIS CATREE);
              END
            END
          END
        END
      END
    END
  END
```

```

        END;
    LSTCODE := CCODE;
    END;

    CS := SVCSCK(CSVC);
    CN := COUNCK(CNAT);
    CM := CT.SELECT(CN,CS);
    IF CM == NONE THEN
        BEGIN
            CM := NEW COMPO(CN,CS);
            CM.INTO(CT.COMPOS);
            END;
        CP := NEW COMPONENT(UOMTAB.FIND(CUM),CPART.SUB(1,1).GETCHAR,
            CMAG.SUB(1,6).GETREAL,CHORIZ.SUB(1,7).GETREAL,
            CVERT.SUB(1,7).GETREAL,COTH.SUB(1,7).GETREAL,
            CMIN.SUB(1,3).GETINT);
        IF CTYP = "C" THEN CM.TEMP := CP
            ELSE IF CTYP = "E" THEN CM.INTL := CP
                ELSE IF CTYP = "W" THEN CM.WARPR := CP
                    ELSE
                        BEGIN
                            OUTTEXT(" -----CATCODE REJECTED--");
                            OUTTEXT(CATIN.IMAGE.SUB(68,4));
                            OUTIMAGE;
                            END;
                        CATIN.INIMAGE;
                        END;
                    CATIN.CLOSE;
                    OUTTEXT(" III. CATCODE TREE COMPLETE");OUTIMAGE;
                    END -- BUILD --;

END ** CATREE **;
```

Figure B-35

k. Class FACTREE.

i. Purpose. The AVL tree for planning factors is implemented by the ROOT CLASS FACTREE. As with CATREE, procedure BUILD accesses the designated file containing the factors and constructs the tree.

ii. Class FACTREE Skeleton.

Parameters and Attributes for Class FACTREE.

CLASS: FACTREE	PREFIX: ROOT
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
FIND	Procedure that searches through the tree structure for an entry that matches the submitted key
BUILD	Constructs the tree. The source file is read and tree entries are created.

Figure B-36

iii. Listing.

Listing of Class FACTREE.

```

ROOT CLASS FACTREE;
BEGIN

REF(FACTOR) PROCEDURE FIND(T); TEXT T;
BEGIN
REF(FACTOR)F; INTEGER I;
F :- FIRST;
IF F == NONE THEN
BEGIN FIND :- NONE;
OUTTEXT(" FACTORS EMPTY");OUTIMAGE;
END
ELSE
WHILE F /= NONE DO
IF T < F.KEY THEN F :- F.LL
ELSE
IF T = F.KEY THEN
BEGIN FIND :- F;F :- NONE; END
ELSE
F :- F.RL;
END ** FIND **;

PROCEDURE BUILD(ECHO,FACTORFILE); TEXT FACTORFILE; BOOLEAN ECHO;
BEGIN
REF(INFILE) FACIN; REF(FACELEM)F; REF(FACTOR)FAC;
REF(FACTORX)FAX; INTEGER FC,FS;
TEXT FLST,FACID,FCAT,FUOM,FRATE,FNAT,FSVC;

```

```

FACIN :- NEW INFILE(FACTORFILE);
FACIN.OPEN(BLANKS(80));
FACIN.INIMAGE;
FLST:- BLANKS(9);
FNAT:-FACIN.IMAGE.SUB(8,1);      FUOM :- FACIN.IMAGE.SUB(18,2);
FSVC:-FACIN.IMAGE.SUB(9,1);      FRATE :- FACIN.IMAGE.SUB(21,10);
WHILE NOT FACIN.ENDFILE DO
  BEGIN
    FACID :- FACIN.IMAGE.SUB(1,6).STRIP;
    FCAT :- FACIN.IMAGE.SUB(11,6).STRIP;
    IF FLST = FACIN.IMAGE.SUB(1,9) THEN
      BEGIN
        IF F /= NONE THEN F :- F.NX :-
          NEW FACELEM(FCAT,FRATE.SUB(1,10).GETREAL,FUOM);
        END
      ELSE
        BEGIN
          IF FNAT = " " THEN
            BEGIN
              FAC :- FIND(FACID);
              IF FAC == NONE THEN
                BEGIN
                  FAC :- NEW FACTOR(FACID);
                  FAC.RANK(THIS FACTREE);
                  F :- FAC.FIRST :- NEW FACELEM(FCAT,
                    FRATE.SUB(1,10).GETREAL,FUOM);
                END
              ELSE
                BEGIN
                  IF FAC IS FACTORX THEN
                    BEGIN
                      OUTTEXT("..FACTOR GENERAL INCONSISTENCY");
                      OUTTEXT(FACIN.IMAGE);OUTIMAGE;
                    END
                  ELSE
                    BEGIN
                      F:-FAC.FIRST;WHILE F.NX /= NONE DO F:-F.NX;
                      F:-F.NX:-NEW FACELEM(FCAT,
                        FRATE.SUB(1,10).GETREAL,FUOM);
                    END;
                  IF ECHO THEN
                    BEGIN OUTTEXT(">FAX:");OUTTEXT(FACIN.IMAGE);
                      OUTIMAGE;END;
                    END
                  ELSE
                    BEGIN
                      FC:=COUNCK(FNAT);
                      FS:=SVCCK(FSVC);
                      IF FC*FS = 0 THEN
                        BEGIN
                          OUTTEXT("..FACTOR..ERROR..");
                          OUTTEXT(FACIN.IMAGE);OUTIMAGE;
                          F :- NONE;
                        END
                      ELSE

```

```

BEGIN
IF ECHO THEN BEGIN OUTTEXT(">FAC:");
                OUTTEXT(FACIN.IMAGE);OUTIMAGE;END;
FAC:-THIS FACTREE.FIND(FACID);
IF FAC == NONE THEN
    BEGIN
        FAC:-NEW FACTORX(FACID,0,0);
        FAC.RANK(THIS FACTREE);
        FAX:-NEW FACTORX(FACID,FC,FS);
        FAX.INTO(FAC QUA FACTORX.VARS);
        F:-FAX.FIRST:-NEW FACELEM(FCAT,
            FRATE.SUB(1,10).GETREAL,FUOM);
    END
ELSE
    IF FAC IS FACTOR THEN
        BEGIN
            OUTTEXT("..FACTOR SPEC INCONSISTENCY ");

            OUTTEXT(FACIN.IMAGE);OUTIMAGE;
            F :- NONE;
            END
        ELSE
            BEGIN
                FAX :- FAC QUA FACTORX.SEL(FC,FS);
                IF FAX == NONE THEN
                    BEGIN
                        FAX :- NEW FACTORX(FACID,FC,FS);
                        FAX.INTO(FAC QUA FACTORX.VARS);
                        F:-FAX.FIRST:-NEW FACELEM(FCAT,
                            FRATE.SUB(1,10).GETREAL,FUOM);
                    END
                ELSE
                    BEGIN
                        F:-FAX.FIRST;
                        WHILE F.NX /= NONE DO F :- F.NX;
                        F:-F.NX:-NEW FACELEM(FCAT,
                            FRATE.SUB(1,10).GETREAL,FUOM);
                    END;
                END;
            END;
        END;
    END;

    FLST := FACIN.IMAGE.SUB(1,9);
    END;
    FACIN.INIMAGE;
    END;
    FACIN.CLOSE;
    OUTTEXT("    IV. FACTOR TREE FINISHED");OUTIMAGE;
    END -- BUILD --;

END ** FACTREE **;

```

Figure B-37

1. Class ECAPTREE.

i. Purpose. ECAPTREE is the ROOT CLASS that structures the engineer unit capability data.

ii. Class ECAPTREE Skeleton.

Parameters and Attributes for Class ECAPTREE.

CLASS: ECAPTREE	PREFIX: ROOT
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
FIND	Procedure that searches through the tree structure for an entry that matches the submitted key
BUILD	Constructs the tree. The source file is read and tree entries are created.

Figure B-38

iii. Listing.

Listing of Class ECAPTREE.

```

ROOT CLASS ECAPTREE;
BEGIN
  REF(ENGRCAP) PROCEDURE FIND(T); TEXT T;
  BEGIN
    REF(ENGRCAP) C; INTEGER I;
    C :- FIRST;
    IF C == NONE THEN
      BEGIN FIND :- NONE;
      OUTTEXT(" CAPABILITY EMPTY");OUTIMAGE;
      END
    ELSE
      WHILE C /= NONE DO
        IF T < C.EUTC THEN C :- C.LL
        ELSE
          IF T = C.EUTC THEN BEGIN FIND :- C;C :- NONE; END
          ELSE
            C :- C.RL;

```



```

        END ** FIND **;

PROCEDURE BUILD(ECHO,ECAPFYL); TEXT ECAPFYL; BOOLEAN ECHO;
BEGIN
    REF(INFILE)CAPABILITY;
    TEXT CAP,CAPUTC,CAPNAT,CAPSV,CH,CV,CO,CSTR;
    INTEGER XIN,XIS,I,J;

    CAP :- BLANKS(80);
    CAPABILITY :- NEW INFILE(ECAPFYL);
    CAPABILITY.OPEN(CAP);
    CAPABILITY.INIMAGE;

    CAPUTC :- CAP.SUB(2,5);    CSTR :- CAP.SUB(28,4);
    CAPNAT :- CAP.SUB(8,1);    CAPSV :- CAP.SUB(10,1);
    CH :-CAP.SUB(33,5);    CV :-CAP.SUB(38,5);    CO :-CAP.SUB(43,5);

    WHILE NOT CAPABILITY.ENDFILE DO
        BEGIN
            IF ECHO THEN BEGIN OUTTEXT(">EUN:");OUTTEXT(CAP);OUTIMAGE;END;
            XIN := COUNCK(CAPNAT);
            XIS := SVCSCK(CAPSV);
            IF XIN = 0 OR XIS = 0 THEN
                BEGIN
                    IF NOT ECHO THEN OUTTEXT(CAP);
                    OUTTEXT("..CAP.. NAT/SVC ERROR -- ");OUTIMAGE;
                    END
                ELSE
                    NEW ENGRCAP(CAPUTC,XIN,XIS,CSTR.SUB(1,4).GETREAL,
CH.SUB(1,5).GETREAL,CV.SUB(1,5).GETREAL,CO.SUB(1,5).GETREAL)
                        .RANK(THIS ECAPTREE);
                    CAPABILITY.INIMAGE;
                END;
            CAPABILITY.CLOSE;
            OUTTEXT("      V. ENGINEER CAPABILITY TREE BUILT.");OUTIMAGE;
            END --CAPBUILD--;

        END ** ECAPTREE **;

```

Figure B-39

m. Class UOMTREE.

i. Purpose. UOMTREE is the ROOT CLASS that defines the AVL tree used for information concerning valid unit of measures associated with facilities and requirements in an application of the model. As with other AVL trees procedure BUILD reads the data in from a designated file.

ii. Class UOMTREE Skeleton.

Parameters and Attributes for Class UOMTREE.

CLASS: UOMTREE	PREFIX: ROOT
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
FIND	Procedure that searches through the tree structure for an entry that matches the submitted key
BUILD	Constructs the tree. The source file is read and tree entries are created.

Figure B-40

iii. Listing.

Listing of Class UOMTREE.

```
ROOT CLASS UOMTREE;
BEGIN
  REF(UNOFMS) PROCEDURE FIND(T); TEXT T;
  BEGIN
    REF(UNOFMS) U; INTEGER I;
    U :- FIRST;
    IF U == NONE THEN
      BEGIN FIND :- NONE;
      OUTTEXT(" UOM EQUIVALENTS EMPTY");OUTIMAGE;
      END
    ELSE
      WHILE U != NONE DO
        IF T < U.UOM THEN U :- U.LL
        ELSE
          IF T = U.UOM THEN BEGIN FIND :- U;U :- NONE; END
          ELSE
            U :- U.RL;
          END ** FIND **;
        END
      END
    END
  END
PROCEDURE BUILD( UOMFYL,ECHO ); TEXT UOMFYL; BOOLEAN ECHO;
BEGIN
  REF(INFILE)UOMEQUIVS;TEXT UBUF;

  UBUF :- BLANKS(80);
  UOMEQUIVS :- NEW INFILE(UOMFYL);
  UOMEQUIVS.OPEN(UBUF);
  UOMEQUIVS.INIMAGE;
```

```

WHILE NOT UOMEQUIVS.ENDFILE DO
  BEGIN
    IF ECHO THEN BEGIN OUTTEXT(">UOM:");OUTTEXT(UBUF);OUTIMAGE;END;
    NEW UNOFMS(UBUF.SUB(1,2),UBUF.SUB(6,2),UBUF.SUB(10,10).GETREAL,
      UBUF.SUB(21,4)).RANK(THIS UOMTREE);
    UOMEQUIVS.INIMAGE;
    END;
  UOMEQUIVS.CLOSE;
  OUTTEXT("      XI. UNIT MEASURE EQUIVALENTS TREE BUILT.");OUTIMAGE;
  END --BUILD--;

END ** UOMTREE **;

```

Figure B-41

n. Class GEOTREE.

i. Purpose. GEOTREE is the ROOT CLASS for an AVL tree of GEOLOCS. Unlike the ROOT CLASSES described above, the GEOTREE is created by an external procedure, BUILDMAP, to enable MAPCELL, REGION, and GEOLOC to be handled in one routine.

ii. Class GEOTREE Skeleton.

Parameters and Attributes for Class GEOTREE.

CLASS: GEOTREE		PREFIX: ROOT	
PARAMETERS: none			
VARIABLE ATTRIBUTES: none			
PROCEDURE ATTRIBUTES:			
FIND		Procedure that searches through the	
		tree structure for an entry that	
		matches the submitted key	
BUILD		Constructs the tree. The source	
		file is read and tree entries are	
		created.	

Figure B-42

iii. Listing.

Listing of Class GEOTREE.

```
ROOT CLASS GEOTREE;
  BEGIN
    REF(GEOLOC) PROCEDURE FIND(T); TEXT T;
    BEGIN
      REF(GEOLOC) C; INTEGER I;
      C :- FIRST;
      IF C == NONE THEN
        BEGIN FIND :- NONE;
        OUTTEXT(" GEOLOCS EMPTY");OUTIMAGE;
        END
      ELSE
        WHILE C != NONE DO
          IF T < C.GEOLAB THEN C :- C.LL
          ELSE
            IF T = C.GEOLAB THEN
              BEGIN
                IF C IS GEOSUB THEN
                  BEGIN
                    C :- C QUA GEOSUB.MAINGEOLOC;
                    IF C != NONE THEN
                      BEGIN FIND :- C;C :- NONE; END
                    ELSE
                      FIND :- NONE;
                    END
                  ELSE
                    BEGIN FIND :- C;C :- NONE; END;
                END
              ELSE
                C :- C.RL;
            END ** FIND **;
          END ** GEOTREE **;
```

Figure B-43

o. Class TGTPREF.

i. Purpose. When an enemy plane attacks an installation, targets are grouped by type to emulate likely targeting strategies. When a group of planes are assembled to attack an installation the analyst can designate primary, secondary, etc. targets types or use the default targeting strategies associated with the installation CLASS. This target priority scheme is accomplished through the TGTPREF objects. Such objects are associated either with installations or missions.

ii. Class TGTPREF Skeleton.

Parameters and Attributes for Class TGTPREF.

CLASS: TGTPREF		PREFIX: none
PARAMETERS:		
NIN	integer	indicates the number of facility groups associated with the installation
VARIABLE ATTRIBUTES:		
VECTOR	real []	indicates how sorties will be allocated against target groups
PROCEDURE ATTRIBUTES: none		

Figure B-44

iii. Listing.

Listing of Class TGTPREF.

```
CLASS TGTPREF(NIN);  
  INTEGER NIN;  
  BEGIN  
    REAL ARRAY VECTOR(1:NIN);  
  END ** TGTPREF **;
```

Figure B-45

p. Class LINKAGE.

i. Purpose. LINKAGE is an internal SIMULA CLASS that is associated with the implementation of two-way lists. Since it is defined by the system its code is not accessible or controlled by the user. ESC has outlined its listing below, as well as give explanations of its attributes as done for other CLASSES. A two-way list is a means of storing data where an object's predecessor and successor in the list is maintained. LINKAGE when used as a prefix to a CLASS provides the pointers and several operations that manipulate them.

ii. Class LINKAGE Skeleton.

Parameters and Attributes for Class LINKAGE.

CLASS: LINKAGE		PREFIX: none
PARAMETERS: none		
VARIABLE ATTRIBUTES:		
SUCC	LINKAGE	pointer to successor (protected)
PRED	LINKAGE	pointer to predecessor (protected)
PROCEDURE ATTRIBUTES:		
SUC	LINK	supplies reference to SUCC
PRED	LINK	supplies reference to PRED

Figure B-46

iii. Listing.

Listing of Class LINKAGE.

```

CLASS LINKAGE;
  BEGIN
    REF(LINKAGE) SUCC, PREDD;

    REF(LINK) PROCEDURE SUC;.....;
    REF(LINK) PROCEDURE PRED;.....;

  END **LINK**;
```

Figure B-47

q. Class HEAD.

i. Purpose. HEAD is an internal SIMULA CLASS that is used to create two-way lists (sometimes referred to as sets or queues). HEAD is prefixed by LINKAGE (see above) and also is not accessible directly by programmers. HEAD performs a similar function as ROOT does for AVL trees -- it is the peg to which objects in the list are attached.

ii. Class HEAD Skeleton.

Parameters and Attributes for Class HEAD.

CLASS: HEAD		PREFIX: LINKAGE
PARAMETERS: none		
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
FIRST	LINK	point to the first object (LINK) in the list
LAST	LINK	points to the last object in list
EMPTY	boolean	returns true is no objects are in the list
CARDINAL	integer	returns number of objects in list
CLEAR		takes each object out of list, and sets their SUC & PRED to NONE

Figure B-48

iii. Listing.

Listing of Class HEAD.

```
LINKAGE CLASS HEAD;  
  BEGIN  
    PROCEDURE CLEAR; .....;  
    REF(LINK) PROCEDURE FIRST; .....;  
    REF(LINK) PROCEDURE LAST; .....;  
    BOOLEAN PROCEDURE EMPTY; .....;  
    INTEGER PROCEDURE CARDINAL; .....;  
  END ** HEAD **;
```

Figure B-49

r. Class ORDLIST.

i. Purpose. As indicated by having HEAD as a CLASS prefix, ORDLIST implements a set that is used to store information about the particular ordnance used in the current application of SEAC. Only one additional attribute has been added to ORDLIST beyond those normally provided by HEAD. To facilitate finding particular ordnance information (see CLASS ORDNANCE) a FIND function has been included. By making it an attribute of ORDLIST, the procedure is associated with the object upon which it will operate, contributing to understanding and correct usage.

ii. Class ORDLIST Skeleton.

Parameters and Attributes for Class ORDLIST.

CLASS: ORDLIST		PREFIX: HEAD
PARAMETERS: none		
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
FIND	ORDNANCE	searches through the list until an ORDNANCE object with a matching ID is found

Figure B-50

iii. Listing.

Listing of Class ORDLIST.

```
HEAD CLASS ORDLIST;
  BEGIN
    REF(ORDNANCE) PROCEDURE FIND(T); TEXT T;
    BEGIN
      REF(ORDNANCE) ORDN; INTEGER I;
      ORDN :- FIRST;
      IF ORDN == NONE THEN
        BEGIN FIND :- NONE;
          OUTTEXT(" ORDNANCE TABLE EMPTY");OUTIMAGE;
        END
      ELSE
        WHILE ORDN /= NONE DO
          IF T = ORDN.ORDID THEN
            BEGIN FIND:-ORDN;ORDN:-NONE;END
            ELSE ORDN:-ORDN.SUC;
          END ** FIND **;
        END ** ORDLIST **;
```

Figure B-51

s. Class LINK.

i. Purpose. LINK is the third internal SIMULA CLASS (along with LINKAGE and HEAD) used extensively within SEAC. Objects from CLASSES prefixed by LINK can be put into lists or sets indicated by objects from CLASS HEAD. The use of sets in SEAC is pervasive. The principal reason is to remove the need to store data in arrays. Typically in FORTRAN programs one will be told that the system will accept 10 plane types, or will track 8 time periods, or will play as many as 100 divisions. Usually one can expect that the reason for the ceiling is because somewhere in the program an array has been defined having maximum dimensions. To a large degree SEAC has avoided such artificial limits by relying extensively on list and trees. While doing so adds some additional storage overhead (e.g., successor and predecessor pointers), in the long run it should provide better memory utilization and access.

ii. Class LINK Skeleton.

Parameters and Attributes for Class LINK.

CLASS: LINK	PREFIX: LINKAGE
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
OUT	removes the object from the set
FOLLOW	places object after named object
PRECEDE	places object before named object
INTO	puts object into designated set

Figure B-52

iii. Listing.

Listing of Class LINK.

```
LINKAGE CLASS LINK;
  BEGIN
    PROCEDURE OUT;.....;
    PROCEDURE INTO(H); REF(HEAD(H));.....;
    PROCEDURE PRECEDE(X); REF(LINKAGE)X; .....;
    PROCEDURE FOLLOW(X); REF(LINKAGE)X;.....;
  END ** LINK **;
```

Figure B-53

t. Class NODE.

i. Purpose. Objects in CLASS NODE are the other components, along with ROOT objects, necessary to implement AVL trees in SEAC. To be useful NODE must be used as a CLASS prefix to a user defined CLASS of objects that represent the information to be structured as a tree. (NODE is used in virtually the same way as LINK, both allow placement of an object in an information structure--a balanced tree in one case, a doubly linked list in the other.)

ii. Class NODE Skeleton.

Parameters and Attributes for Class NODE.

CLASS: NODE		PREFIX: LINK
PARAMETERS:		
TEST	VIRTUAL	establishes sort key and checks whether key match is made
DUMP	VIRTUAL	user provided procedure that prints information about the entry
VARIABLE ATTRIBUTES:		
BAL	integer	each NODE object maintains an indicator of its branch's balance
LL	NODE	points to a NODE defined by TEST to precede the current NODE
RL	NODE	points to a NODE defined by TEST to succeed the current NODE
PROCEDURE ATTRIBUTES:		
RANK		inserts the node into a tree identified by a designated ROOT.
SEARCH		is the principal routine in the AVL tree implementation. It first locates the correct position of the candidate node in the tree. If an insertion occurs SEARCH will re-balance the tree if necessary.
TRAVERSE		moves through the tree in key order printing specified information

Figure B-54

iii. Listing.

Listing of Class NODE.

```
LINK CLASS NODE;
VIRTUAL : INTEGER PROCEDURE TEST;PROCEDURE DUMP;
BEGIN
REF(NODE)LL,RL;
INTEGER BAL;

PROCEDURE RANK(X); REF(ROOT)X;
BEGIN
REF(NODE)T,NEXT;
INTEGER I;BOOLEAN H;
IF X == NONE THEN OUTTEXT("ROOT == NONE")
ELSE
BEGIN
T:-X.FIRST;
IF T == NONE THEN X.FIRST:- THIS NODE
ELSE
SEARCH(T,H);
END;
END *** RANK ***;

PROCEDURE SEARCH(P,H); REF(NODE) P; BOOLEAN H;
BEGIN
INTEGER IC;
REF(NODE)P1,P2;

IC :- TEST(P);
IF IC < 0 THEN
BEGIN
IF P.LL == NONE THEN
BEGIN
P.LL :- THIS NODE;BAL :-0;H :- TRUE;
END
ELSE
SEARCH(P.LL,H);
IF H THEN
BEGIN
IF P.BAL = 1 THEN
BEGIN P.BAL:=0;H:=FALSE;END
ELSE
IF P.BAL = 0 THEN
P.BAL := -1
ELSE
BEGIN
P1 :- P.LL;
IF P1.BAL = -1 THEN
BEGIN
P.LL :- P1.RL;
P1.RL :- P;
P.BAL := 0;
P :- P1;
END
END
END
END
```

```

ELSE
BEGIN
P2 :- P1.RL;
P1.RL :- P2.LL;
P2.LL :- P1;
P.LL :- P2.RL;
P2.RL :- P;
IF P2.BAL = -1 THEN
    P.BAL := +1
ELSE
    P.BAL := 0;
IF P2.BAL = +1 THEN
    P1.BAL := -1
ELSE
    P1.BAL := 0;
P :- P2;
END;
P.BAL := 0;
H := FALSE;
END;

END;
ELSE
IF IC = +1 THEN
BEGIN
IF P.RL == NONE THEN
BEGIN
P.RL :- THIS NODE;
BAL := 0;
H := TRUE;
END
ELSE
SEARCH(P.RL,H);
IF H THEN
BEGIN
IF P.BAL = -1 THEN
BEGIN P.BAL:=0;H:=FALSE;END
ELSE
IF P.BAL = 0 THEN
P.BAL := +1
ELSE
BEGIN
P1 :- P.RL;
IF P1.BAL = +1 THEN
BEGIN
P.RL :- P1.LL;
P1.LL :- P;
P.BAL := 0;
P :- P1;
END
ELSE
BEGIN
P2 :- P1.LL;
P1.LL :- P2.RL;

```

```

P2.RL :- P1;
P.RL :- P2.LL;
P2.LL :- P;
IF P2.BAL = +1 THEN
    P.BAL := -1
ELSE
    P.BAL := 0;
IF P2.BAL = -1 THEN
    P1.BAL := +1
ELSE
    P1.BAL := 0;
P :- P2;
END;
P.BAL := 0;
H := FALSE;
END;
END;
END
ELSE
BEGIN
H := FALSE;
END;
END -- SEARCH --;

PROCEDURE TRAVERSE;
BEGIN
IF LL/=NONE THEN LL.TRAVERSE;
DUMP;
IF RL/=NONE THEN RL.TRAVERSE;
END -- TRAVERSE --;

END *** NODE ***;

```

Figure B-55

u. Class CATCOD.

i. Purpose. SEAC determines engineering requirements in terms of the amount and type of facilities that are required by friendly forces. These needs are then translated into actual manhour requirements by looking up how much effort must be expended to build (repair, maintain) the facility. This information comes from the different service facility component systems. CATCOD objects are not only in the AVL tree associated with CATREE but also have an internal data structure of their own. Each object contains a list of objects (COMPOs) that indicate which nation-service has entries for the catcode. Each of these objects can have three other objects (COMPONENT) that provide details on the characteristics of the facility and the engineer effort necessary to construct or repair it.

ii. Class CATCOD Skeleton.

Parameters and Attributes for Class CATCOD.

CLASS: CATCOD		PREFIX: CATCOD
PARAMETERS:		
CATLAB	text	the four character JCS catcode identifier of a facility category
CATNAM	text	the title associated with the facility category
VARIABLE ATTRIBUTES:		
COMPOS	HEAD	is a list of different country and service entries that have facility components in the category
PROCEDURE ATTRIBUTES:		
TEST		sorts entries in the tree by comparing CATLABs
SELECT	COMPO	searches through a list of country service entries for a match
DUMP		formats information about the object to be printed during TRANSVERSAL

Figure B-56

iii. Listing.

Listing of Class CATCOD.

```
NODE CLASS CATCOD(CATLAB,CATNAM);
  VALUE CATLAB,CATNAM;TEXT CATLAB,CATNAM;
  BEGIN
    REF(HEAD)COMPOS;

    INTEGER PROCEDURE TEST(T);REF(CATCOD) T;
      TEST := IF CATLAB < T.CATLAB THEN -1
              ELSE IF CATLAB > T.CATLAB THEN 1 ELSE 0;

    REF(COMPO) PROCEDURE SELECT(N,S);INTEGER N,S;
      BEGIN
        REF(COMPO) R;
        R :- COMPOS.FIRST;
        WHILE R /= NONE DO
          IF N = R.NAT AND S = R.SVC THEN
            BEGIN SELECT :- R;R :- NONE;END
          ELSE R :- R.SUC;
        END -- SELECT --;

    PROCEDURE DUMP;
      BEGIN
        REF(COMPO)CMP;
        OUTTEXT(CATLAB);SETPOS(10);OUTTEXT(CATNAM);
        CMP :- COMPOS.FIRST;
        WHILE CMP /= NONE DO
          BEGIN
            SETPOS(35);OUTINT(CMP.NAT,3);OUTINT(CMP.SVC,3);
            IF CMP.INTL /= NONE THEN
              BEGIN SETPOS(50);OUTTEXT("INITIAL"); END;
            IF CMP.TEMP /= NONE THEN
              BEGIN SETPOS(60);OUTTEXT("TEMPORARY"); END;
            IF CMP.WARPR /= NONE THEN
              BEGIN SETPOS(70);OUTTEXT("WAR DAMAGE"); END;
            OUTIMAGE;
            CMP :- CMP.SUC;
          END;
        END -- DUMP --;

    COMPOS :- NEW HEAD;
    END ** CATCOD **;
```

Figure B-57

v. Class FACTOR.

i. Purpose. Much of SEAC is involved with determining the amount of facilities necessary to support an OPLAN. Some requirements are dictated by special analysis of the theater and the plan; most requirements, however, result from the use of planning factors. Units, installation types, equipment, and people can all generate facility needs. SEAC combines all these factors in the AVL tree defined by CLASS FACTREE. The search key in the tree is the item

label that induces the facility requirement. The CLASS type, equipment identifier, unit type code (UTC), or a special key (e.g., "PEOPLE") are possible values for KEY. Each FACTOR (and FACTORX) object has a list of facility items (FACELEMs) associated with it that indicate what the requirements are. One point worth noting is that FACTOR objects have no nation or service identifier but are considered applicable for all.

ii. Class FACTOR Skeleton.

Parameters and Attributes for Class FACTOR.

CLASS: FACTOR		PREFIX: NODE
PARAMETERS:		
KEY	text	the factor predicate
VARIABLE ATTRIBUTES:		
FIRST	FACELEM	points to the first facility factor for the factor predicate
PROCEDURE ATTRIBUTES:		
TEST	integer	compares given key to FACTOR key
DUMP		prints out the FACTOR-FACELEM data

Figure B-58

iii. Listing.

Listing of Class FACTOR.

```
NODE CLASS FACTOR(KEY);
  VALUE KEY;TEXT KEY;
  BEGIN
    REF(FACELEM) FIRST;

    INTEGER PROCEDURE TEST(T);REF(FACTOR)T;
      TEST := IF KEY < T.KEY THEN -1
        ELSE IF KEY> T.KEY THEN 1 ELSE 0;

    PROCEDURE DUMP;
      BEGIN
        REF(FACELEM)FAXEL;
        OUTTEXT(KEY);OUTIMAGE;
        FAXEL :- FIRST;
        WHILE FAXEL -/- NONE DO
          BEGIN SETPOS(10);
            OUTTEXT(FAXEL.CODE);OUTFIX(FAXEL.RATE,2,10);
            OUTCHAR(' ');OUTTEXT(FAXEL.UOM); OUTIMAGE;
            FAXEL :- FAXEL.NX;
          END;
        END -- DUMP --;
      END ** FACTOR **;
```

Figure B-59

w. Class FACTORX.

i. Purpose. FACTORX is a subclass of FACTOR that is used when factors must be distinguish by nation and service. Most factors fall in this category. The initial occurrence of a factor of this type gives rise to an entry in the tree with country and service variables set to 0. Another FACTORX object is created, with the given country and service values, and placed in the VARS list.

ii. Class FACTORX Skeleton.

Parameters and Attributes for Class FACTORX.

CLASS: FACTORX		PREFIX: FACTOR
PARAMETERS:		
COUN	integer	designates nationality
SRVC	integer	designates service
VARIABLE ATTRIBUTES:		
VAR	HEAD	contains the list of FACTORX distinguished by country, service
PROCEDURE ATTRIBUTES:		
DUMP		prints out "all" factor data
SEL	FACTOR	picks the object in VAR that has the required country, service

Figure B-60

iii. Listing.

Listing of Class FACTORX.

```

FACTOR CLASS FACTORX(COUN,SRVC);
  INTEGER COUN,SRVC;
  BEGIN
    REF(HEAD)VAR;

    REF(FACTOR)PROCEDURE SEL(C,S);INTEGER C,S;
      BEGIN
        REF(FACTORX)FX;
        FX:-VAR.FIRST;
        WHILE FX -/= NONE DO
          IF C=FX.COUN AND S=FX.SRVC THEN
            BEGIN SEL :- FX;FX :- NONE;END
          ELSE FX :- FX.SUC;
        END -- SEL --;

    PROCEDURE DUMP;
      BEGIN
        REF(FACELEM)EL;REF(FACTORX)FX;
        FX :- VAR.FIRST;
        WHILE FX -/= NONE DO
          BEGIN
            OUTTEXT(FX.KEY);OUTINT(FX.COUN,10);OUTINT(FX.SRVC,10);OUTIMAGE;
            EL :- FX.FIRST;
            WHILE EL -/= NONE DO
              BEGIN SETPOS(10);
                OUTTEXT(EL.CODE);

```

```

OUTFIX(EL.RATE,2,10);OUTCHAR(' ');OUTTEXT(EL.UOM);
OUTIMAGE;
EL :- EL.NX;
END;
FX :- FX.SUC;
END;
END -- DUMP --;

VARS:-NEW HEAD;
END ** FACTORX **;

```

Figure B-61

x. Class ENGRCAP.

i. Purpose. Determining engineer capability in SEAC is relatively straightforward. The manhours associated with engineer units in a particular place are added together (after some modification) to determine the total capability. The source of those manhours is the engineer capability tree (ECAPTREE), which has a list of valid engineer units identified by (i.e. with search key) UTC. Objects from CLASS ENGRCAP are the nodal entries in the tree that contain that information.

ii. Class ENGRCAP Skeleton.

Parameters and Attributes for Class ENGRCAP.

CLASS: ENGRCAP		PREFIX: NODE
PARAMETERS:		
EUTC	text	unit type code (engineer unit only)
NATION	integer	designates nationality
SERVICE	integer	designates service
HRZ	real	horizontal manhours
VRT	real	vertical manhours
OTH	real	other manhours
STRENGTH	real	nominal strength of unit
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
TEST	integer	
DUMP		prints unit data

Figure B-62

iii. Listing.

Listing of Class ENGRCAP.

```
NODE CLASS ENGRCAP(EUTC,NATION,SERVICE,STRENGTH,HRZ,VRT,OTH);
  VALUE EUTC;TEXT EUTC;INTEGER NATION,SERVICE;
  REAL HRZ,VRT,OTH,STRENGTH;
  BEGIN

  INTEGER PROCEDURE TEST(T);REF(ENGRCAP)T;
    TEST := IF EUTC < T.EUTC THEN -1
           ELSE IF EUTC > T.EUTC THEN 1 ELSE 0;

  PROCEDURE DUMP;
    BEGIN
      SETPOS(10);
      OUTTEXT(EUTC);OUTFIX(STRENGTH,0,5);
      OUTINT(NATION,2);OUTINT(SERVICE,2);
      OUTFIX(HRZ,2,10);OUTFIX(VRT,2,10);OUTFIX(OTH,2,10);
      OUTIMAGE;
      END -- DUMP --;
  END ** ENGRCAP **;
```

Figure B-63

y. Class UNOFMS.

i. Purpose. With hundreds of different facilities and factor entries indicating the size or amount of particular facilities it is clear that some attention must be paid to measurement units to assure consistency. When a requirement is given in miles and the facility components are in feet, objects from this CLASS are checked to see if a conversion can be made. In addition to consistency, these objects are also used in war damage calculations. If facility components are measured in production terms (e.g., kilo-volts), it is necessary to translate this value into a dimensional value used in the target damage logic. Getting equivalent translations is certainly not exact, but an estimate is necessary to permit damage calculations.

ii. Class UNOFMS Skeleton.

Parameters and Attributes for Class UNOFMS.

CLASS: UNOFMS		PREFIX: NODE
PARAMETERS:		
UOM	text	unit of measure (e.g., sf, mi)
EQVMS	text	equivalent unit of measure
COEF	real	conversion factor from UOM to EQVMS
DAMCRIT	text	default damage criteria e.g., EMDA
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
TEST	integer	
DUMP		prints entry information

Figure B-64

iii. Listing.

Listing of Class UNOFMS.

```

NODE CLASS UNOFMS(UOM, EQVMS, COEF, DAMCRIT);
  VALUE UOM,EQVMS,DAMCRIT;TEXT UOM,EQVMS,DAMCRIT;REAL COEF;
  BEGIN

  INTEGER PROCEDURE TEST(T);REF(UNOFMS)T;
    TEST := IF UOM < T.UOM THEN -1
           ELSE IF UOM > T.UOM THEN 1 ELSE 0;

  PROCEDURE DUMP;
    BEGIN
      SETPOS(10);OUTTEXT(UOM);      SETPOS(20);OUTTEXT(EQVMS);
      SETPOS(30);OUTFIX(COEF,3,10); SETPOS(43);OUTTEXT(DAMCRIT);
      OUTIMAGE;
      END -- DUMP --;
  END ** UNOFMS **;
```

Figure B-65

z. Class COMPO.

i. Purpose. Within each CATCOD object is a list of service & nation entries which have facilities identified by that JCS category code. These COMPO objects can contain component information for initial and temporary facility standards, or war damage repair requirements to repair temporary facilities (see COMPONENT CLASSES). Manipulation of these standards or damage

components plays an important part in the model. The model currently will only schedule new construction using initial standard facilities. If the user does not include an initial COMPONENT the model interprets that as a policy decision to build no new facilities of that component type.

ii. Class COMPO Skeleton.

Parameters and Attributes for Class COMPO.

CLASS: COMPO		PREFIX: LINK	
PARAMETERS:			
NAT	integer	indicates nationality	
SVC	integer	indicates service	
VARIABLE ATTRIBUTES:			
INTL	COMPONENT	points to initial std component	
TEMP	COMPONENT	points to temporary std component	
WARPR	COMPONENT	points to damage repair component	

PROCEDURE ATTRIBUTES: none			

Figure B-66

iii. Listing.

Listing of Class COMPO.

```
LINK CLASS COMPO(NAT,SVC);
  INTEGER NAT,SVC;
  BEGIN
  REF(COMPONENT)INTL,TEMP,WARPR;
  END ** COMPO **;
```

Figure B-67

a. Class GEOLOC.

i. Purpose. Most references to locations in SEAC use the geographical location code (GEOLOC). They are the means to identify installations in the theater. GEOLOCS are used pervasively in the model and it was therefore necessary to have an efficient referencing scheme, to save computer time. An AVL tree (see GEOTREE) was selected as the most convenient means to accomplish this. Within each object is a list of the installations associated with that GEOLOC, distinguished by service and country.

ii. Class GEOLOC Skeleton.

Parameters and Attributes for Class GEOLOC.

CLASS: GEOLOC		PREFIX: NODE
PARAMETERS:		
GEOLAB	text	designates the geographic location code
VARIABLE ATTRIBUTES:		
CELL	MAPCELL	points to the mapcell in which the GEOLOC belongs
SITES	HEAD	the set of installations in the GEOLOC
PROCEDURE ATTRIBUTES:		
TEST	integer	compares GEOLABs for placement
DUMP		prints out GEOLOC information

Figure B-68

iii. Listing.

Listing of Class GEOLOC.

```
NODE CLASS GEOLOC(GEOLAB);
  VALUE GEOLAB; TEXT GEOLAB;
  BEGIN
    REF(MAPCELL) CELL;
    REF(HEAD)SITES;

    INTEGER PROCEDURE TEST(T);REF(GEOLOC) T;
      TEST := IF GEOLAB < T.GEOLAB THEN -1
              ELSE IF GEOLAB > T.GEOLAB THEN 1 ELSE 0;

    PROCEDURE DUMP;
      BEGIN
        REF(INSTALLATION)S;
        SETPOS(10);OUTTEXT(GEOLAB);
        S :- SITES.FIRST;
        WHILE S /= NONE DO
          BEGIN
            SETPOS(20); OUTTEXT(S.INAME);OUTTEXT(" IS A ");
            OUTTEXT(INSTLIS.SUB(1+4*(S.INDX-1),4));OUTIMAGE;
            S :- S.SUC;
          END;
        END -- DUMP --;

    SITES :- NEW HEAD;
  END ** GEOLOC **;
```

Figure B-69

b. Class GEOSUB.

i. Purpose. This CLASS was created to enable different GEOLOCs to be grouped together and treated as a single installation. (It is somewhat analogous to the base complexing done in the CESPg.) Objects of this type are identified when the GEOTREE is being built by having a "CMPX" entered in its installation field, and then identifying what the parent GEOLOC will be for the duration of the model's execution.

ii. Class GEOSUB Skeleton.

Parameters and Attributes for Class GEOSUB.

CLASS: GEOSUB	PREFIX: GEOLOC
PARAMETERS:	
MAINGEoloc GEOLOC	point to the GEOLOC that this GEOLOC is considered part of
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
DUMP	prints out identity of this GEOLOC and its parent GEOLOC

Figure B-70

iii. Listing.

Listing of Class GEOSUB.

```
GEOLOC CLASS GEOSUB(MAINGEoloc);
REF(GEOLOC) MAINGEoloc;
BEGIN
PROCEDURE DUMP;
    BEGIN SETPOS(10);
    OUTTEXT(GEOLAB);SETPOS(25);
    OUTTEXT("SUBCOMPLEX OF GEOLOC ");
    OUTTEXT(MAINGEoloc.GEOLAB);OUTIMAGE;
    END -- DUMP --;

SITES :- MAINGEoloc.SITES;
CELL :- MAINGEoloc.CELL;
END ** GEOSUB **;
```

Figure B-71

c. Class PROPERTY.

i. Purpose. Much of the model revolves around determining what facilities are at or required at installations. REALTYTREE instances at each installation establish AVL trees that keep track of: how much of a facility is on the installation; what amount of the facility is required according to people, unit, equipment, etc. factors; and as a result of shortfalls what amount of the facility is currently to be or is being built. The result might be equivalent to maintaining a property book. PROPERTY objects are the nodes in the tree structure, and correspond to individual category codes (not FACILITY subCLASSES). Each day the installation is queried to check the status

of facility requirements. On d-Day, before deployment begins, a special routine, INITIALIZE, is invoked. It compares the facility requirements at the installation (using only in-theater units) with the assets. An assumption made, however, is that all H-hour requirements are met even there are no assets indicated on the ASSET file. After that all succeeding facility assessments are done through routine ASSESS, which schedules new construction when shortfalls occur.

ii. Class PROPERTY Skeleton.

Parameters and Attributes for Class PROPERTY.

CLASS: PROPERTY		PREFIX: NODE	
PARAMETERS:			
CTCODE	text	the 4 character JCS catcode for	
		this property	
VARIABLE ATTRIBUTES:			
REQD	real	amount of facility category req'd	
UNDERWAY	real	amount of facility category being	
		built	
ONHAND	real	amount of existing facility category	
CATPTR	CATCOD	points to reference cat code	
CMP	COMPO	points to applicable COMPO object	
UNMS	UNOFMS	points to unit of meas being used	
PROCEDURE ATTRIBUTES:			
TEST	integer		
DUMP			
XREF	boolean	finds CMP and UNMS for this object	
NEEDS		increases amount of requirements	
SHOWS		increase amount of onhand assets	
INITIALIZE		at day 0 a check is made to see if	
		facility assets are adequate for	
		in theater unit/base requirements.	
		If not onhand assets are increased.	
ASSESS		determines whether and which new	
		facilities are needed at the	
		installaiton	

Figure B-72

iii. Listing.

Listing of Class PROPERTY.

```
NODE CLASS PROPERTY(CTCODE);
  VALUE CTCODE;TEXT CTCODE;
  BEGIN
  REAL REQD,UNDERWAY,ONHAND;
  REF(CATCOD)CATPTR;REF(COMPO)CMP;REF(UNOFMS)UNMS;

  INTEGER PROCEDURE TEST(R);REF(PROPERTY)R;
    TEST := IF CTCODE < R.CTCODE THEN -1
            ELSE IF CTCODE > R.CTCODE THEN 1 ELSE 0;

  PROCEDURE DUMP;
    BEGIN
    SETPOS(10); OUTTEXT(CTCODE);
    OUTFIX(REQD,1,20); OUTFIX(ONHAND,1,10); OUTFIX(UNDERWAY,1,10);
    OUTIMAGE;
    END;

  BOOLEAN PROCEDURE XREF(INSTL); REF(INSTALLATION)INSTL;
    BEGIN
    IF CATPTR == NONE THEN
      BEGIN
      OUTTEXT("....PROP/XREF..MISSING CATCOD ");OUTTEXT(CTCODE);
      OUTIMAGE;
      END
    ELSE
      BEGIN
      CMP:-CATPTR.SELECT(INSTL.COUN,INSTL.SRVC);
      IF CMP == NONE AND INSTL.COUN <> XUS THEN
        CMP:-CATPTR.SELECT(XUS,INSTL.SRVC);
      IF CMP == NONE THEN
        BEGIN
        OUTTEXT("....PROP/XREF..MISSING COMPO..");OUTTEXT(CTCODE);
        OUTIMAGE;
        END
      ELSE
        IF CMP.TEMP /= NONE THEN
          BEGIN UNMS :- CMP.TEMP.UOM; XREF := TRUE; END
        ELSE
          IF CMP.INTL /= NONE THEN
            BEGIN UNMS :- CMP.INTL.UOM; XREF := TRUE; END;
          END;
        END -- XREF --;

  PROCEDURE NEEDS(AMT,UM); REAL AMT; TEXT UM;
    IF UM = UNMS.UOM THEN REQD := REQD + AMT
    ELSE
      IF UM = UNMS.EQVMS THEN REQD := REQD + UNMS.COE*AMT
      ELSE
        BEGIN
```

```

        OUTTEXT("...PROP/NEEDS..UOM MISMATCH..");OUTTEXT(CTCODE);
        OUTCHAR(' ');OUTTEXT(UM);
        OUTIMAGE;
        END;

PROCEDURE SHOWS(AMT,UM); REAL AMT; TEXT UM;
    IF UM = UNMS.UOM THEN ONHAND := ONHAND + AMT
    ELSE
        IF UM = UNMS.EQVMS THEN ONHAND := ONHAND + UNMS.COEFF*AMT
        ELSE
            BEGIN
                OUTTEXT("...PROP/ASSETS..UOM MISMATCH..");OUTTEXT(CTCODE);
                OUTCHAR(' ');OUTTEXT(UM);
                OUTIMAGE;
                END;

PROCEDURE INITIALIZE(INSTL);REF(INSTALLATION)INSTL;
    BEGIN REF(COMPONENT) CMPNT;CHARACTER STD;REF(FACILITY) F;REAL STRT,SZ;

    IF LL=NONE THEN LL QUA PROPERTY.INITIALIZE(INSTL);
    IF REQD < ONHAND THEN STRT:= ONHAND
        ELSE STRT := ONHAND := REQD;
    IF STRT > 0.0 THEN
        BEGIN
            IF CMP.TEMP /= NONE THEN
                BEGIN CMPNT:-CMP.TEMP; STD:=XTSTD;END
            ELSE IF CMP.INTL /= NONE THEN
                BEGIN CMPNT:-CMP.INTL;STD:=XISTD;END;
            IF CMPNT /= NONE THEN
                BEGIN
                    IF CMPNT.PART = XFRAC THEN
                        BEGIN
                            F :- CATOFAC(THIS PROPERTY,CMP,STRT,CTCODE,STD);
                            IF F /= NONE THEN
                                BEGIN INSTL.GETS(F,CTCODE); F.BUILT:=TRUE;END;
                            END
                        ELSE
                            IF CMPNT.PART = XWHL THEN
                                BEGIN
                                    SZ:=IF STRT > CMPNT.SIZE THEN CMPNT.SIZE
                                        ELSE STRT;
                                    F :- CATOFAC(THIS PROPERTY,CMP,SZ,CTCODE,STD);
                                    IF F /= NONE THEN
                                        BEGIN
                                            STRT := STRT - SZ;
                                            INSTL.GETS(F,CTCODE);
                                            F.BUILT := TRUE;
                                            WHILE STRT > 0.0 DO
                                                BEGIN
                                                    IF SZ > STRT THEN SZ := STRT;
                                                    F :- CATOFAC(THIS PROPERTY,CMP,
                                                                SZ,CTCODE,STD);
                                                    INSTL.GETS(F,CTCODE);
                                                    F.BUILT := TRUE;

```

```

                                STRT := STRT - SZ;
                                END;
                                END;
                                END
                                ELSE
                                BEGIN
                                OUTTEXT("....PROP/INIT..PARTITIONING NOT
                                                KNOWN..");
                                OUTTEXT(CTCODE); OUTIMAGE;
                                END;
                                END
                                ELSE
                                BEGIN
                                OUTTEXT("....PROP/INIT..CONSTRUCTION STD NOT KNOWN..");
                                OUTTEXT(CTCODE); OUTIMAGE;
                                END;
                                END;
                                IF RL=-NONE THEN RL QUA PROPERTY.INITIALIZE(INSTL);
                                END -- INITIALIZE --;

PROCEDURE ASSESS(INSTL);REF(INSTALLATION)INSTL;
BEGIN REF(COMPONENT) CMPNT;CHARACTER STD;REF(FACILITY) F;
REAL SHORTAGE,MULT;
IF LL=-NONE THEN LL QUA PROPERTY.ASSESS(INSTL);
SHORTAGE := REQD-(ONHAND+UNDERWAY);
IF SHORTAGE > 0.0 THEN
    BEGIN
    IF CMP.INTL /= NONE THEN
        BEGIN
        CMPNT:-CMP.INTL;        STD:=XISTD;
        IF UNMS.UOM = CMPNT.UOM.UOM THEN MULT := 1.0
        ELSE
            IF UNMS.EQVMS = CMPNT.UOM.UOM THEN
                BEGIN
                SHORTAGE:=UNMS.COEF*SHORTAGE;
                MULT := 1/UNMS.COEF;
                END
            ELSE
                CMPNT :- NONE;
            END;
        IF CMPNT /= NONE THEN
            BEGIN
            IF CMPNT.PART = XFRAC THEN
                BEGIN
                F :- CATOFAC(THIS PROPERTY,CMP,SHORTAGE,CTCODE,STD);
                IF F /= NONE THEN
                    BEGIN
                    F.CONSTR(INSTL.COUN,INSTL.SRVC,
                                INSTL.INDX,INSTL.JOBS);
                    UNDERWAY := UNDERWAY + SHORTAGE*MULT;
                    INSTL.GETS(F,CTCODE);
                    END;
                END
            ELSE

```

```

        IF CMPNT.PART = XWHL THEN
            BEGIN
                WHILE SHORTAGE >= CMPNT.SIZE DO
                    BEGIN
                        F :- CATOFAC(THIS PROPERTY,CMP,
                                    CMPNT.SIZE,CTCODE,STD);
                        F.CONSTR(INSTL.COUN,INSTL.SRVC,
                                INSTL.INDX,INSTL.JOBS);
                        SHORTAGE:=SHORTAGE-CMPNT.SIZE;
                        UNDERWAY := UNDERWAY + CMPNT.SIZE*MULT;
                        INSTL.GETS(F,CTCODE);
                    END;
                END
            ELSE
                BEGIN
                    OUTTEXT("      ....PROP/ASSESS..PARTITION
                                ERROR..");
                    OUTTEXT(CTCODE);
                    OUTIMAGE;
                END;
            END
        ELSE
            BEGIN
                OUTTEXT("      ....PROP/ASSESS..NO COMPONENT (UOM/STD)..");
                OUTTEXT(CTCODE); OUTIMAGE;
            END;
        END;
    END;
    IF RL/=NONE THEN RL QUA PROPERTY.ASSESS(INSTL);
    END -- ASSESS --;

    CATPTR :- CATAB.FIND(CTCODE);
    END ** PROPERTY **;

```

Figure B-73

d. Class ENGRUNIT.

i. Purpose. Engineer capability in SEAC is a direct measure of the manhours provided by the various engineer units that operate in the theater. ENGRUNIT objects represent these engineer units. Each day at every installation and in every region where there are engineer units (see ENGRPOOL), SEAC evaluates the effective capability of the assigned units and adds it to a local pool of available engineer horizontal, vertical, and other manhours. The calculation is performed daily in case there have been changes in unit location or effectiveness. This addresses both the phase in period where units are arriving in theater, and a general availability factor that wraps up travel, sickness, and casualties. Presently SEAC does not explicitly calculate casualty losses (although the structure of SEAC could easily be adapted to do so). One of the strengths of the model is its ability to explicitly manipulate engineer units. This provides an internal double check against moving units that don't exist, complements the detailed theater representation and also provides an opportunity to consider service and skill distinctions that might otherwise be impossible. (Some of the advantages are admittedly prospective,

but, since SEAC envisions future extensions, within the objectives of the model's design.)

ii. Class ENGRUNIT Skeleton.

Parameters and Attributes for Class ENGRUNIT.

CLASS: ENGRUNIT		PREFIX: LINK
PARAMETERS:		
UTCMSTR	ENGRCAP	points to ECAPB file reference UTC
STRENGTH	real	manpower strength from TPFDD
TIMTHT	real	day unit arrives in theater
VARIABLE ATTRIBUTES:		
HCAP	real	horizontal hours for UTCMSTR
VCAP	real	vertical " " "
OCAP	real	other " " "
ADJUTC	real	adjusts capability based on: time in theater, strength, and general availability.
PROCEDURE ATTRIBUTES:		
MUSTER		makes daily assessment of capabil- ity of unit and adds hours to pool.

Figure B-74

iii. Listing.

Listing of Class ENGRUNIT.

```
LINK CLASS ENGRUNIT(UTCMSTR,STRENGTH,TIMTHT);
REF(ENGRCAP)UTCMSTR;REAL STRENGTH,TIMTHT;
BEGIN
REAL HCAP,VCAP,OCAP,ADJUTC;

PROCEDURE MUSTER(H,V,O,DRP,S); NAME H,V,O,S; REAL H,V,O,S;BOOLEAN DRP;
BEGIN
ADJUTC:=IF TIME < TIMTHT + 9 THEN
ENGBLDUP(UTCMSTR.NATION,ENTIER(TIME)-TIMTHT) *
ENGRAVL(UTCMSTR.NATION)
ELSE
ENGRAVL(UTCMSTR.NATION);
H := H + HCAP*ADJUTC; V := V + VCAP*ADJUTC; O := O + OCAP*ADJUTC;
S := S + ADJUTC * ( HCAP + VCAP + OCAP );
IF DRP THEN
BEGIN SETPOS(53);OUTFIX(HCAP,1,8);OUTFIX(VCAP,1,8);
OUTFIX(OCAP,1,8);OUTFIX(ADJUTC,3,7);OUTCHAR(' ');
OUTTEXT(UTCMSTR.EUTC); OUTIMAGE;
END;
END -- MUSTER --;

| ROOM IS LEFT FOR CASUALTY, MOPP, AND SKILL SUBSTITUTION DATA AND CODE;

BEGIN
IF TIMTHT < 0.0 THEN TIMTHT := -10.0;
ADJUTC := STRENGTH / UTCMSTR.STRENGTH;
HCAP := ADJUTC * UTCMSTR.HRZ;
VCAP := ADJUTC * UTCMSTR.VRT;
OCAP := ADJUTC * UTCMSTR.OTH;
END;
END ** ENGRUNIT **;
```

Figure B-75

e. Class ORDNANCE.

i. Purpose. This is the general CLASS under which all ordnance used to arm enemy units or planes must be defined. It is left to ORDNANCE subCLASSES to provide specific details.

ii. Class ORDNANCE Skeleton.

Parameters and Attributes for Class ORDNANCE.

CLASS: ORDNANCE		PREFIX: LINK
PARAMETERS:		
ORDID	text	the name of the ordnance
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES: none		

Figure B-76

iii. Listing.

Listing of Class ORDNANCE.

```
LINK CLASS ORDNANCE(ORDID); VALUE ORDID;TEXT ORDID; ;
```

Figure B-77

f. Class BOMB.

i. Purpose. This ORDNANCE subCLASS is used to represent the essential characteristics of enemy air delivered bombs. (Presently only "dumb" bombs are used.) Two attributes give radii of fragmentation and blast damage for the bomb. Calculating accurate bomb damage to primary and collateral targets is complicated and requires a level of representational detail inappropriate to that used in SEAC. SEAC attempts to generate a representative damage estimate that considers the threat, the ordnance, the installation target, and the interplay of facility and ordnance. (The current form of SEAC damage calculation was largely determined by form of data supplied by PACAF for the Korean theater.) Since most targets are buildings, damage is usually proportional to the square feet of either blast or fragmentation damage. Bombs dropped on runways, buildings, POL tanks , etc., invoke special damage calculation routines.

ii. Class Skeleton.

Parameters and Attributes for Class BOMB.

CLASS: BOMB		PREFIX: ORDNANCE
PARAMETERS:		
RFRAG	real	fragmentation radius against initial standard facility
RBLST	real	blast radius against temporary standard facility
DUDRATE	real	bomb reliability
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
EFFECT	real	selects the appropriate radius

Figure B-78

iii. Listing.

Listing of Class BOMB.

```

ORDNANCE CLASS BOMB(RFRAG,RBLST,DUDRATE);
  REAL RFRAG,RBLST,DUDRATE;
  BEGIN

  REAL PROCEDURE EFFECT(TYPSTD); CHARACTER TYPSTD;
    EFFECT := IF TYPSTD = XISTD THEN RFRAG
              ELSE IF TYPSTD = XTSTD THEN RBLST
              ELSE 0.0;

  END ** BOMBS **;
```

Figure B-79

g. Class ROCKET.

i. Purpose. ROCKET is an ordanace subCLASS used to represent the effects of unguided rockets. In the initial version of SEAC, damage is not calculated for rocket attacks because of the limited capability of 57mm rockets to cause significant structural damage.

ii. Class ROCKET Skeleton.

Parameters and Attributes for Class ROCKET.

CLASS: ROCKET	PREFIX: ORDANCE
PARAMETERS:	
DUDRATE real	rocket reliability factor
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES: none	

Figure B-80

iii. Listing.

Listing of Class ROCKET.

```
ORDNANCE CLASS ROCKET(DUDRATE);REAL DUDRATE; ;
```

Figure B-81

h. Class DEMOL.

i. Purpose. Ordnance from this CLASS is used to represent the demolition ordnance or explosive material that insurgent or commando units would carry to carry out sabotage in the COMMZ (see SPFUNIT). Attributes represent material factors that were extracted from FM 5-25 for damage radii against wood and masonry targets. Also included is an estimation of the amount of debris that would have to be cleared from an attack on a tunnel that was sabotaged. (FM 5-25 suggested the relative difficulty of causing major damage to a tunnel, especially one that will probably be guarded.)

ii. Class DEMOL Skeleton.

Parameters and Attributes for Class DEMOL.

CLASS: DEMOL		PREFIX: ORDNANCE
PARAMETERS:		
MTLFACTIM	real	material factor from FM 5-25 for breaching timber
MTLFACMAS	real	material factor from FM 5-25 for breaching good masonry
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES:		
EFFECT	real	damage radius for demolition and target interaction
DEBRIS	real	calculates amount of debris (cubic measure) from tunnel attack

Figure B-82

iii. Listing.

Listing of Class DEMOL

```

ORDNANCE CLASS DEMOL(MTLFACTIM,MTLFACMAS);
REAL MTLFACTIM,MTLFACMAS;
BEGIN
REAL PROCEDURE EFFECT(TYPSTD,LBS); CHARACTER TYPSTD;REAL LBS;
BEGIN |FORMULA ADAPTED FROM FM 5-25,SEC-V,BREACH RADII;
|RADIUS := CUBE ROOT OF LBS TNT/(MATL FAC * TAMP FAC);
EFFECT := EXP(.33 * ( LN(LBS)-
(LN(IF TYPSTD = XISTD THEN MTLFACTIM ELSE MTLFACMAS)
+ LN(3.6) ) ) );
END -- EFFECT --;

REAL PROCEDURE DEBRIS(LBSEX);REAL LBSEX;
|THIS IS ALSO AN ADAPTATION FROM FM 5-25
WHERE MATERIAL FAC = .27 (ROCK >> 7 FT) &
TAMPING FAC = 1.8;
DEBRIS:=(2/3)*PI*LBSEX/(.27*1.8);
END ** DEMOL **;
```

Figure B-83

i. Class AIRCRAFT.

i. Purpose. This CLASS is used to define the types of aircraft that the enemy will use to attack the COMMZ. (Originally calculations were intended to actually calculate time to and from target to measure turn around capacity. The uncertainties of routes and level of attack caused this feature to be dropped and enemy planes were assumed to be capable of meeting mission schedules found in intelligence estimates.)

ii. Class AIRCRAFT Skeleton.

Parameters and Attributes for Class AIRCRAFT.

CLASS: AIRCRAFT		PREFIX: LINK
PARAMETERS:		
ID	text	identification label of plane type
RANGE	real	range (not used)
SPEED	real	speed (not used)
VARIABLE ATTRIBUTES:		
PAYLOAD	HEAD	ordnance that planes of this type are configured to carry
PID	integer	identifies plane as fighter or fighter/bomber
PROCEDURE ATTRIBUTES: none		

Figure B-84

iii. Listing.

Listing of Class AIRCRAFT.

```
LINK CLASS AIRCRAFT(ID,RANGE,SPEED);
VALUE ID; TEXT ID; REAL RANGE,SPEED;
BEGIN REF(HEAD)PAYLOAD;INTEGER PID; PAYLOAD:-NEW HEAD;END;
```

Figure B-85

j. Class MUNITION.

i. Purpose. Objects from this CLASS are placed in the PAYLOAD set in AIRCRAFT objects to define the ordnance that is being carried by that plane type.

ii. Class MUNITION Skeleton.

Parameters and Attributes for Class MUNITION.

CLASS: MUNITION		PREFIX: LINK
PARAMETERS:		
TYPORD	ORDNANCE	points to the ordnance type
NUMBER	real	indicates how many are carried
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES: none		

Figure B-86

iii. Listing.

Listing of Class MUNITION.

```
LINK CLASS MUNITION(TYPORD,NUMBER);  
REF(ORDNANCE)TYPORD;REAL NUMBER; ;
```

Figure B-87

k. Class PLANE.

i. Purpose. SEAC models enemy air attacks through explicit representation of enemy threat assets. A PLANE object is created for every plane in the enemy's order of battle that can deliver munitions against COMMZ installations. PLANES carry the MUNITIONS that are dropped (fired) at FACILITY targets that result in facility damage. This combination of objects from different CLASSES is handled in the internal PLANE procedure ATTACK. Actually PLANE objects are grouped together to attack various targets by the mission generation process (see PROCESS MISGEN).

ii. Class PLANE Skeleton.

Parameters and Attributes for Class PLANE.

CLASS: PLANE		PREFIX: LINK
PARAMETERS:		
TYPE	AIRCRAFT	identifies the type of aircraft
VARIABLE ATTRIBUTES:		
FACTGT	FACILITY	the selected target
DROP	integer	amount of ordnance dropped / pass
AIMPOINT	integer	index of target set
REPAIRTIME	real	time when plane will be repaired
AMMO	real	size of ordnance load
MUN	MUNITION	points to (each) ordnance type being carried
PROCEDURE ATTRIBUTES:		
ATTACK		this routine plays the attack of the plane against a target

Figure B-88

iii. Listing.

Listing of Class PLANE.

```

LINK CLASS PLANE (TYPE);
REF(AIRCRAFT) TYPE;
BEGIN
REF(FACILITY) FACTGT; INTEGER DROP, AIMPOINT;
REAL REPAIRTIME, AMMO; REF(MUNITION) MUN;

PROCEDURE ATTACK(TGTSITE); REF(INSTALLATION) TGTSITE;
BEGIN
TGTSITE.SORTIESRECVD := TGTSITE.SORTIESRECVD + 1;
MUN := TYPE.PAYLOAD.FIRST;
IF ATKLOG THEN
    BEGIN OUTTEXT("....."); OUTTEXT(TYPE.ID); OUTINT(AIMPOINT, 4);
    OUTCHAR(' '); OUTTEXT(TGTSITE.INAME); OUTCHAR('- ');
    END;
WHILE MUN /= NONE DO
    BEGIN
    AMMO := MUN.NUMBER;
    IF MUN.TYPORD IS ROCKET THEN
        TGTSITE.ROCHITS := TGTSITE.ROCHITS + AMMO
    ELSE
        IF MUN.TYPORD IS BOMB THEN

```



```

WHILE AMMO > 0 DO
  BEGIN
    DROP :- IF AMMO < RELRATE THEN AMMO ELSE RELRATE;
    AMMO :- AMMO - DROP;
    FACTGT :- TGTSITE.TGTSELECT(AIMPOINT);
    IF FACTGT -/- NONE THEN
      BEGIN
        IF ATKLOG THEN OUTTEXT(FACTGT.PRP.CTCODE);
        IF DRAW(FACHIT(AIMPOINT),UFHIT) THEN
          FACTGT.DAMAGE(MUN.TYPORD,DROP,TGTSITE.DAMLIST)
        ELSE
          IF AIMPOINT = 7 THEN
            BEGIN
              IF DRAW(FACHIT(6),UFHIT) THEN
                BEGIN
                  FACTGT :- TGTSITE.TGTSELECT(6);
                  IF FACTGT IN SURFACE THEN
                    FACTGT.DAMAGE(MUN.TYPORD,DROP,TGTSITE.DAMLIST);
                END;
              END
            ELSE
              IF DRAW(FACHIT(4),UFHIT) THEN
                BEGIN
                  FACTGT :- TGTSITE.TGTSELECT(4);
                  IF FACTGT IN TRANSPOR THEN
                    FACTGT.DAMAGE(MUN.TYPORD,DROP,TGTSITE.DAMLIST);
                END
              ELSE
                IF TGTSITE IS AIRBASE THEN
                  BEGIN
                    IF DRAW(FACHIT(9),UFHIT) THEN
                      BEGIN
                        FACTGT :- TGTSITE.TGTSELECT(5);
                        IF FACTGT IN UTILITY THEN
                          FACTGT.DAMAGE(MUN.TYPORD,
                            DROP,TGTSITE.DAMLIST);
                      END
                    ELSE
                      IF DRAW(FACHIT(9),UFHIT) THEN
                        BEGIN
                          FACTGT:-TGTSITE.TGTSELECT(9);
                          IF FACTGT IN UTILITY THEN
                            FACTGT.DAMAGE(MUN.TYPORD,
                              DROP,TGTSITE.DAMLIST);
                        END;
                      END
                    END
                  ELSE
                    IF DRAW(FACHIT(5),UFHIT) THEN
                      BEGIN
                        FACTGT :- TGTSITE.TGTSELECT(5);
                        IF FACTGT IN UTILITY THEN
                          FACTGT.DAMAGE(MUN.TYPORD,
                            DROP,TGTSITE.DAMLIST);
                        END;
                      END
                    ELSE
                      IF DRAW(FACHIT(5),UFHIT) THEN
                        BEGIN
                          FACTGT :- TGTSITE.TGTSELECT(5);
                          IF FACTGT IN UTILITY THEN
                            FACTGT.DAMAGE(MUN.TYPORD,
                              DROP,TGTSITE.DAMLIST);
                        END;
                      END
                    END
                  END
                END
              END
            END
          END
        END
      END
    END
  END

```

```

        END;
      END;
    MUN :- MUN.SUC;
    IF ATKLOG THEN OUTIMAGE;
  END;
END -- ATTACK --;

END ** PLANE **;

```

Figure B-89

1. Class NKAIRBASE.

i. Purpose. As part of the play of threat air attacks, SEAC can keep track of the planes that are found at different enemy airfields. Planes are placed into four different categories: fighters, bombers, those being repaired, and those that are not operational (but not because of war damage). Whenever the mission generation process calls for a new attack from an airbase SEAC checks to see which planes are available from those that have been repaired and those that are declared operationally ready. (NB: the importance of this CLASS is somewhat less than originally conceived for two reasons. Damage to enemy bases that would destroy planes on the ground or deny use of the base is not played. And the location of the base relative to the target is also waived as a constraint. The latter is the reason why the range and speed attributes of PLANE are not used in SEAC. A different scenario might require that consideration of damage and enemy airbase capacity be taken into account.)

ii. Class NKAIRBASE Skeleton.

Parameters and Attributes for Class NKAIRBASE.

CLASS: NKAIRBASE		PREFIX: LINK
PARAMETERS:		
ID	text	name of the airbase
VARIABLE ATTRIBUTES:		
BOMBERS	HEAD	set of BMBR plane objects
FIGHTERS	HEAD	set of FTR plane objects
REPAIR	HEAD	PLANEs in for repair
NOTRDY	HEAD	PLANEs found not operationally ready
PROCEDURE ATTRIBUTES:		
PREPARE		determines which planes have been repaired, and which ones are operationally ready and available for attacks

Figure B-90

iii. Listing.

Listing of Class NKAIRBASE.

```
LINK CLASS NKAIRBASE(ID);
VALUE ID;TEXT ID;
BEGIN
REF(HEAD)BOMBERS,FIGHTERS,REPAIR,NOTRDY;

PROCEDURE PREPARE(OPRDY); REAL OPRDY;
BEGIN
REF(PLANE)PL,TEMP;
INTEGER NB,NF,NOB,NOF,NR;
FOR PL :- NOTRDY.FIRST WHILE PL /= NONE DO
    IF PL.TYPE IS BMBR THEN PL.INTO(BOMBERS)
    ELSE PL.INTO(FIGHTERS);
PL :- REPAIR.FIRST;
WHILE PL /= NONE DO
    IF PL.REPAIRTIME LE TIME THEN
        BEGIN
            TEMP :- PL.SUC;
            IF PL.TYPE IS BMBR THEN
                PL.INTO(BOMBERS)
            ELSE
                PL.INTO(FIGHTERS);
```

```

        PL :- TEMP;
        END
        ELSE
        BEGIN NR := NR +1; PL :- PL.SUC; END;
    PL :- BOMBERS.FIRST;
    WHILE PL /= NONE DO
        IF DRAW(OPRDY,URDY) THEN
            BEGIN NB:=NB+1; PL:-PL.SUC; END
            ELSE
            BEGIN
                TEMP :- PL.SUC; PL.INTO(NOTRDY);
                NOB:=NOB+1; PL:-TEMP;
            END;
        PL :- FIGHTERS.FIRST;
        WHILE PL /= NONE DO
            IF DRAW(OPRDY,URDY) THEN
                BEGIN NF:=NF+1; PL:-PL.SUC; END
                ELSE
                BEGIN
                    TEMP :- PL.SUC; PL.INTO(NOTRDY);
                    NOF:=NOF+1; PL:-TEMP;
                END;
            SETPOS(13); OUTTEXT(ID);
            OUTINT(NB,7); OUTCHAR('/'); OUTINT(NOB,4);
            OUTINT(NF,5); OUTCHAR('/'); OUTINT(NOF,4);
            OUTINT(NR,10); OUTIMAGE;
            END ** PREP ** ;

REPAIR :- NEW HEAD;
BOMBERS :- NEW HEAD;
NOTRDY :- NEW HEAD;
FIGHTERS :- NEW HEAD;
END **NKAIRBASE ** ;

```

Figure B-91

m. Class SPFUNIT.

i. Purpose. This CLASS is used to represent the enemy units or groups that will be inserted into the rear and used against COMMZ targets. Just as there is a PLANE object for each enemy aircraft, so there is an SPFUNIT object for each enemy infiltrated unit. Unlike PLANE objects, however, SPFUNIT objects are not recycled--when a unit expends the ordnance it carries it is no longer considered by SEAC.

ii. Class SPFUNIT Skeleton.

Parameters and Attributes for Class SPFUNIT.

CLASS: SPFUNIT		PREFIX: LINK

PARAMETERS:		
TORG	TEAMORG	the type of the unit
LOCALE	MAPCELL	where the unit is

VARIABLE ATTRIBUTES: none		

PROCEDURE ATTRIBUTES: none		

Figure B-92

iii. Listing.

Listing of Class SPFUNIT.

```
LINK CLASS SPFUNIT(TORG, LOCALE); REF(MAPCELL) LOCALE; REF(TeamORG) TORG; ;
```

Figure B-93

n. Class TEAMORG.

i. Purpose. The ground threat is defined in terms of number and placement of enemy units in the COMMZ. The model is only interested in calculating the damage that these units can cause and what that translates into in terms of engineer requirements. Therefore SEAC is only interested in how much demolition materiel that the unit carries; what the unit does to kill troops and disrupt rear operations is not within the scope of the model. TEAMORG objects define the types of teams that will be inserted into the COMMZ. (TEAMORG is to SPFUNIT what AIRCRAFT is to PLANE.)

ii. Class TEAMORG Skeleton.

Parameters and Attributes for Class TEAMORG.

CLASS: TEAMORG		PREFIX: LINK
PARAMETERS:		
TID	text	identification label of threat team/organization
VARIABLE ATTRIBUTES:		
ARMS	HEAD	list of ordnance and amounts that team carries that can damage facilities
PROCEDURE ATTRIBUTES:		
SABOTAGE	boolean	attacks MSR & LOC facilities
ATK		carries out attacks against 'soft' targets (buildings,utilities) on an installation

Figure B-94

iii. Listing.

Listing of Class TEAMORG

```

LINK CLASS TEAMORG(TID);VALUE TID; TEXT TID;
  BEGIN
    REF(HEAD)ARMS;

    PROCEDURE ATK(INSTL);REF(INSTALLATION)INSTL;
      BEGIN
        REF(FACILITY)FACTGT;REF(MUNITION)M;REAL DEMLBS,CHRG;
        IF ATKLOG THEN BEGIN OUTTEXT(INSTL.INAME);OUTCHAR(' ');END;
        INSTL.RCATK := INSTL.RCATK + 1;
        M := ARMS.FIRST;
        WHILE M /= NONE DO
          BEGIN
            IF M.TYPORD IS DEMOL THEN
              BEGIN
                DEMLBS := M.NUMBER;
                WHILE DEMLBS > 0 DO
                  BEGIN
                    CHRG:=IF LBSTGT<DEMLBS THEN LBSTGT ELSE DEMLBS;
                    FACTGT:-RANDPICK(INSTL.FBLDG);
                    IF FACTGT /= NONE THEN
                      BEGIN
                        IF ATKLOG THEN OUTTEXT(FACTGT.PRP.CTCODE);

```

```

FACTGT.DAMAGE(M.TYPORD,CHRG,INSTL.DAMLIST);
DEMLBS:=DEMLBS-CHRG;
CHRG:=IF LBSGTGT<DEMLBS THEN LBSGTGT
        ELSE DEMLBS;

END;
FACTGT:-IF DEMLBS > 0 THEN RANDPICK(INSTL.FUTIL)
        ELSE NONE;
IF FACTGT /= NONE THEN
    BEGIN
        IF ATKLOG THEN OUTTEXT(FACTGT.PR.PCTCODE);
        FACTGT.DAMAGE(M.TYPORD,CHRG,INSTL.DAMLIST);
        DEMLBS:=DEMLBS-CHRG;
        END;
    IF DEMLBS - M.NUMBER THEN DEMLBS := 0;
    END;
END;
M :- M.SUC;
END;
END --ATK--;

BOOLEAN PROCEDURE SABOTAGE(MC);REF(MAPCELL)MC;
BEGIN
    REF(FACILITY)MSRTGT;REF(MUNITION)M;
    IF ATKLOG THEN OUTTEXT(" MAPCELL ");
    MSRTGT :- IF MC.PIPELINE=-NONE THEN RANDPICK(MC.PIPELINE)
        ELSE IF MC.RRS=-NONE THEN RANDPICK(MC.RRS)
        ELSE IF MC.ROADS=-NONE THEN RANDPICK(MC.ROADS)
        ELSE NONE;
    IF MSRTGT == NONE THEN SABOTAGE:=FALSE
    ELSE
        BEGIN
            M :- ARMS.FIRST;
            WHILE M /= NONE DO
                BEGIN
                    IF M.TYPORD IS DEMOL THEN
                        MSRTGT.DAMAGE(M.TYPORD,M.NUMBER,MC.DMGLIST);
                    M :- M.SUC;
                    END;
                SABOTAGE := TRUE;
            END;
        END
    END --SABO--;

ARMS:-NEW HEAD;
END ** TEAMORG **;
```

Figure B-95

o. Class FACILITY.

i. Purpose. The prefix CLASS FACILITY defines the basic data and procedural attributes for all the different facilities categories represented in SEAC. Requirements are expressed in terms of amount of needed facilities. Needed facilities are then translated into work tasks which can be performed by available engineer units. FACILITY objects can also be damaged by enemy air or ground attacks. (Maintenance requirements would be easy to generate but have not been in this implementation.) It is important to note that FACILITY is used as a prefix to the five subCLASSES: BUILDING, SURFACE, PETRO, UTILITIES, and TRANSP0. These in turn are used as prefixes to the actual CLASSES for which objects are generated. (This structural hierarchy used to represent facilities in SEAC requires an understanding of the SIMULA CLASS. ANNEX A introduces this and other special features of SIMULA, and references are also given there for more in depth discussions of SIMULA.)

ii. Class Skeleton.

Parameters and Attributes for Class FACILITY

CLASS: FACILITY		PREFIX: LINK
PARAMETERS:		
SIZE	real	size of the facility
PRP	PROPERTY	points to PROPERTY entry for this facility category
CMPO	COMPO	points to
STD	character	'I' - initial, 'T' - temporary
VIRTUAL:		
DAMAGE		calculates how damage is to be assessed against the type of facility (default is BUILDING)
REPAIR		determines what the damage means in terms of repair hours and whether a TASK must be created
VARIABLE ATTRIBUTES:		
WRKTSK	TASK	points to worktask associated with constructing this FACILITY
COMPNT	COMPONENT	points to the COMPONENT used for this object
DAMREP	HITS	points to the object which accumulates damage against the FACILITY
DMGRT	real	fraction of FACILITY damaged
FACINDX	integer	indexes FACILITY-sub-sub-CLASSes
BUILT	boolean	'true' if building is constructed
PROCEDURE ATTRIBUTES:		
ERROR		error message routine
CONSTR		generates the TASK that enters a construction project
DAMAGE		see VIRTUAL section
FINISHED		this is called when construction is completed
MAINTENANCE		n/a
SELRPRCMP	COMPONENT	how repair time is estimated is a function of construction standard and whether a damage component has been included in COMPO
CKUOM	boolean	makes sure 'units' match or are accounted for by conversion

Figure B-96

iii. Listing.

Listing of Class FACILITY

```
LINK CLASS FACILITY(SIZE,PRP,CMPO,STD);
REF(PROPERTY)PRP;REAL SIZE;REF(COMPO)CMPO;CHARACTER STD;
VIRTUAL : PROCEDURE DAMAGE,REPAIR;
BEGIN
REF(TASK)WRKTSK; REF(COMPONENT)CMPNT; REF(HITS)DAMREP;
REAL DMGRT; INTEGER FACINDX; BOOLEAN BUILT;

PROCEDURE ERROR(T1,T2);TEXT T1,T2;
  BEGIN SETPOS(10);
  OUTTEXT("...FACILITY ERROR...");
  OUTTEXT(T1);OUTTEXT(T2);
  OUTIMAGE;
  END --ERROR --;
PROCEDURE CONSTR(NAT,SRVC,LOCX,JOBQ);
REF(HEAD)JOBQ;INTEGER NAT,LOCX,SRVC;
BEGIN
REF(POLICY) PRIO;REAL MULT;
PRIO :- RATE(NAT,'C',LOCX,FACINDX);
IF PRIO -/- NONE THEN
  BEGIN
  MULT :- PRIO.SHARE*SIZE/CMPNT.SIZE;
  WRKTSK :- NEW TASK(PRIO.PRIORP,NAT,SRVC,MULT*CMPNT.HORZ,
    MULT*CMPNT.VERT,MULT*CMPNT.OTH,
    CMPNT.MINDAY,THIS FACILITY,'C');
  WRKTSK.ADDTO(JOBQ);
  ENTERQT(WRKTSK,NAT);
  END
  ELSE
  ERROR("POLICY MISSING ",PRP.CTCODE);
  END -- CONSTR --;
PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
REF(ORDNANCE)ORDTYP;REAL ORDAMT;REF(HEAD)DAMLST;
BEGIN
REF(UNOFMS)MS;REF(COMPONENT)CT;REAL DAM;
MS :- CMPNT.UOM;
IF ATKLOG THEN OUTTEXT(".FAC/DMG.");
IF MS.EQVMS = " " THEN
  BEGIN
  IF MS.DAMCRIT = "EMDL" THEN
    DAM:=-BLAST(MS.DAMCRIT,EMDTAB(FACINDX),STD,ORDTYP,ORDAMT)
  ELSE
    DAM:=-BLAST(MS.DAMCRIT,SIZE,STD,ORDTYP,ORDAMT);
  END
  ELSE
  IF MS.DAMCRIT = "EMDA" THEN
    DAM:=-BLAST(MS.DAMCRIT,SIZE*MS.COEF,STD,ORDTYP,ORDAMT)
  ELSE
    DAM:=-BLAST(MS.DAMCRIT,SIZE*MS.COEF,STD,ORDTYP,ORDAMT);
  IF ATKLOG THEN BEGIN OUTFIX(DAM,1,7); OUTIMAGE; END;
  IF DAM < 0 THEN
```

```

BEGIN
  IF DAMREP == NONE THEN
    BEGIN
      DAMREP:=-NEW HITS(THIS FACILITY,DAM);
      DAMREP.INTO(DAMLST);
    END
    ELSE
      DAMREP.EXTENT:=-DAMREP.EXTENT+DAM;
    END;
END -- DAMAGE --;
PROCEDURE FINISHED;
BEGIN
  IF PRP /= NONE THEN
    BEGIN
      PRP.UNDERWAY:=-PRP.UNDERWAY-SIZE;
      PRP.ONHAND:=-PRP.ONHAND+SIZE;
      BUILT := TRUE;
    END;
    WRKTSK:-NONE;
  END --FINISHED--;
PROCEDURE MAINTENANCE;
BEGIN
  END -- MAINT --;
REF(COMPONENT) PROCEDURE SELRPRCMP;
  SELRPRCMP :- IF STD = XISTD THEN CMPO.INTL
                ELSE IF CMPO.WARPR == NONE THEN CMPO.TEMP
                ELSE CMPO.WARPR;

BOOLEAN PROCEDURE CKUOM(CPT,TOTDAM);REF(COMPONENT) CPT;REAL TOTDAM;
BEGIN
  IF CPT.UOM /= CMPNT.UOM THEN
    BEGIN
      IF CMPNT.UOM.EQVMS = CPT.UOM.UOM THEN
        DMGRT := TOTDAM * CMPNT.UOM.COEF/CPT.SIZE
      ELSE
        BEGIN
          DMGRT := 0.0;
          ERROR(CPT.UOM.UOM,CMPNT.UOM.UOM);
        END;
      END
    ELSE
      DMGRT := TOTDAM / CPT.SIZE;
    CKUOM := IF DMGRT > 0.0 THEN TRUE ELSE FALSE;
  END -- CKUOM --;

CMPNT :- IF STD= XISTD THEN CMPO.INTL ELSE CMPO.TEMP;
END ** FACILITY **;

```

Figure B-97

p. Class SURFACE.

i. Purpose. This prefix CLASS is used for CLASSES: RUNWAY, PAVEMENT, and HARDSTAND. These FACILITYs are similar in that craters are the means by

which damage is generated and repaired. There can be many craters to repair and on RUNWAYS such craters may have different priorities (to simulate attaining a minimum operational strip).

ii. Class SURFACE Skeleton.

Parameters and Attributes for Class SURFACE.

CLASS: SURFACE	PREFIX: FACILITY
PARAMETERS: none	
VARIABLE ATTRIBUTES:	
CRATERS integer	counts craters to assess priority
PROCEDURE ATTRIBUTES:	
DAMAGE	damage created as craters
REPAIR	separate repair TASKs are generated for each crater

Figure B-98

iii. Listing.

Listing of Class SURFACE.

```

FACILITY CLASS SURFACE;
BEGIN
  INTEGER CRATERS;

  PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
    REF(ORDNANCE) ORDTYP; REAL ORDAMT;REF(HEAD)DAMLST;
    BEGIN
      IF ATKLOG THEN OUTTEXT(" .SUR/DAM. ");
      IF ORDTYP IS BOMB THEN
        BEGIN
          IF DAMREP == NONE THEN
            BEGIN
              DAMREP:-NEW HITS(THIS SURFACE,0.0);
              DAMREP.INTO(DAMLST);
            END;
          WHILE ORDAMT > 0.0 DO
            BEGIN
              IF NOT DRAW(ORDTYP QUA BOMB.DUDRATE,UDUD) THEN
                BEGIN
                  DAMREP.EXTENT:=DAMREP.EXTENT+1.0;
                  IF ATKLOG THEN OUTTEXT(" CRTR ");
                END;
              ORDAMT := ORDAMT - 1.0;
            END;
          END;
        END;
      END;
    END;
  END;

```

```

        END;
    END;
END -- DAMAGE --;

PROCEDURE REPAIR(EXTENT,RC,RS,LOCX,JOBQ);
    INTEGER RC,RS,LOCX;REAL EXTENT;REF(HEAD)JOBQ;
    BEGIN
        REF(POLICY)PRI;INTEGER CRTPRIO,P;REF(COMPONENT)CT;
        CRATERS := CRATERS + EXTENT;
        PRI:-RATE(RC,'D',LOCX,FACINDX);
        IF PRI -/= NONE THEN
            BEGIN
                CT:-CMPO.WARPR;
                IF CT == NONE THEN
                    ERROR("CRATER RR MISSING",NOTEXT)
                ELSE
                    BEGIN
                        CRTPRIO := PRI.PRIGRP;
                        IF THIS SURFACE IS RUNWAY THEN
                            BEGIN
                                WHILE EXTENT > 0 DO
                                    BEGIN
                                        P := CRTPRIO -1 + DISCRETE(CRTRSPLIT,UHOLE);
                                        IF P<1 OR XPRIO<P THEN P := XPRIO;
                                        NEW TASK(P,RC,RS,CT.HORZ,CT.VERT,CT.OTH,
                                            CT.MINDAY,THIS FACILITY,'W').ADDTO(JOBQ);
                                        EXTENT := EXTENT -1;
                                    END;
                                END
                            END
                        ELSE
                            WHILE EXTENT > 0 DO
                                BEGIN
                                    NEW TASK(CRTPRIO,RC,RS,CT.HORZ,CT.VERT,CT.OTH,
                                        CT.MINDAY,THIS FACILITY,'W').ADDTO(JOBQ);
                                    EXTENT := EXTENT -1;
                                END;
                            END
                        END;
                    END;
                END;
            END;
        END;
    END -- REPAIR --;
END ** SURFACE **;

```

Figure B-99

q. Class RUNWAY.

i. Purpose. This is the subCLASS of SURFACE for category codes "111", runways.

ii. Listing.

Listing of Class RUNWAY.

```
SURFACE CLASS RUNWAY;      | 111 ;  
  BEGIN FACINDX := 1; END ** RUNWAY **;
```

Figure B-100

r. Class PAVEMENT.

i. Purpose. This is the subCLASS of SURFACE for category codes: "112", taxiways , "113", parking aprons, "116", aircraft pads, and "153", cargo handling areas.

ii. Listing.

Listing of Class PAVEMENT.

```
SURFACE CLASS PAVEMENT;    | 112 113 116 153;  
  BEGIN FACINDX := 2; END ** PAVEMENT **;
```

Figure B-101

s. Class HARDSTAND.

i. Purpose. This is the subCLASS of SURFACE for category code "852", hardstands or other paved or stabilized areas.

ii. Listing.

Listing of Class HARDSTAND.

```
SURFACE CLASS HARDSTAND;   | 852 ;  
  BEGIN FACINDX := 19; END ** HARDSTAND **;
```

Figure B-102

t. Class BUILDING.

i. Purpose. This CLASS normally will represent the majority of FACILITY objects represented during execution of the model. BUILDING subCLASSES are: COMO, OPNS, REVETMENT, PIER, SHOP, STORAGE, MEDICAL, ADMIN, and QTRS. A general repair procedure is provided within this CLASS, but may be overridden by subCLASS preemption.

ii. Class BUILDING Skeleton.

Parameters and Attributes for Class BUILDING.

CLASS: BUILDING		PREFIX: FACILITY
PARAMETERS: none		
VARIABLE ATTRIBUTES:		
RPRTSK	TASK	points to the repair TASK
CUMDAM	real	the amount of damage
PROCEDURE ATTRIBUTES:		
REPAIR		updates damage amount and either creates or changes RPRTSK

Figure B-103

iii. Listing.

Listing of Class BUILDING.

```

FACILITY CLASS BUILDING;
BEGIN
  REF(TASK)RPRTSK; REAL CUMDAM;

PROCEDURE REPAIR(EXTENT,RC,RS,LOCX,JOBQ);
  INTEGER RC,RS,LOCX;REF(HEAD)JOBQ;REAL EXTENT;
  BEGIN
    REF(POLICY) PRI;REF(COMPONENT)CT;
    IF BUILT THEN
      BEGIN
        INSPECT RPRTSK WHEN TASK DO
          CUMDAM := CUMDAM * ((DUR-PROGRESS)/DUR) + EXTENT
        OTHERWISE
          CUMDAM := CUMDAM + EXTENT;
        PRI := -RATE(RC, 'D', LOCX, FACINDX);
        IF PRI /= NONE THEN
          BEGIN
            IF CUMDAM > SIZE * SUPRFICL(FACINDX,
              IF STD= XISTD THEN 1 ELSE 2) THEN
              BEGIN
                IF CUMDAM > SIZE * STRUCT(FACINDX,
                  IF STD= XISTD THEN 1 ELSE 2) THEN
                  BEGIN
                    PRP.ONHAND:=PRP.ONHAND-SIZE;
                    IF RPRTSK /= NONE THEN RPRTSK.OUT;
                    OUT;
                    END
                  ELSE

```

```

      BEGIN
      CT:-SELRPRCMP;
      IF CKUOM(CT,CUMDAM) THEN
        BEGIN
          IF RPRTSK == NONE THEN
            BEGIN
              RPRTSK:-NEW TASK(PRI.PRIOGRP,RC,RS,
                DMGRT*CT.HORZ,DMGRT*CT.VERT,DMGRT*CT.OTH,
                CT.MINDAY,THIS FACILITY,'W');
              RPRTSK.ADDTO(JOBQ);
            END
          ELSE
            RPRTSK.REVISE(DMGRT,CT);
          END;
        END;
      END;
    END;
  END
  ELSE
  BEGIN
    INSPECT WRKTSK DO PROGRESS := ENTIER(PROGRESS*CUMDAM/SIZE);
    CUMDAM:=0.0;
  END;
END -- REPAIR --;

END ** BUILDING **;

```

Figure B-104

u. Class COMO

i. Purpose. This is the subCLASS of BUILDING for category codes "131", communications buildings, and "133", navigation aids.

ii. Listing.

Listing of Class COMO.

```

BUILDING CLASS COMO;      | 131 133 ;
  BEGIN FACINDX := 6; END ** COMO **;

```

Figure B-105

v. Class OPNS.

i. Purpose. This is the subCLASS of BUILDING for category codes "141", operational buildings, and "156", cargo handling facilities.

ii. Listing.

Listing of Class OPNS.

```
BUILDING CLASS OPNS;      | 141 156;  
  BEGIN FACINDX := 7; END ** OPNS **;
```

Figure B-106

w. Class REVETMENT.

i. Purpose. This is the subCLASS of BUILDING for category codes "149", revetments.

ii. Listing.

Listing of Class REVETMENT.

```
BUILDING CLASS REVETMENT; | 149 ;  
  BEGIN FACINDX := 8; END ** REVETMENT **;
```

Figure B-107

x. Class PIER.

i. Purpose. This is the subCLASS of BUILDING for category codes: "151", piers, "152", wharfs, "159", other waterfront operational, and "163", moorings.

ii. Listing.

Listing of Class PIER.

```
BUILDING CLASS PIER;      | 151 152 159 163 ;  
  BEGIN FACINDX := 9; END ** PIER **;
```

Figure B-108

y. Class SHOP.

i. Purpose. This is the subCLASS of BUILDING for category codes: "211", aircraft maintenance, "213", ship maintenance, "214", automotive maintenance, "215", weapon maintenance, "216", ammunition maintenance, "217", electronic maintenance, "218", miscellaneous maintenance, and "219", real property maintenance.

ii. Listing.

Listing of Class SHOP.

```
BUILDING CLASS SHOP;      | 211 213 214 215 216 217 218 219 ;  
BEGIN FACINDX := 10; END ** SHOP **;
```

Figure B-109

z. Class STORAGE.

i. Purpose. This is the subCLASS of BUILDING for category codes: "421", depot ammunition storage, "425", open ammunition storage, "431", depot cold storage, "432", cold storage, "441", depot covered storage, and "442", other covered storage.

ii. Listing.

Listing of Class STORAGE.

```
BUILDING CLASS STORAGE;   | 421 425 431 432 441 442 ;  
BEGIN FACINDX := 11; END ** STORAGE **;
```

Figure B-110

aa. Class MEDICAL.

i. Purpose. This is the subCLASS of BUILDING for category codes: "510", hospital, "540", dental clinics, "550", dispensaries, and "560", other medical.

ii. Listing.

Listing of Class MEDICAL.

```
BUILDING CLASS MEDICAL;   | 510 540 550 560 ;  
BEGIN FACINDX := 12; END ** MEDICAL **;
```

Figure B-111

bb. Class ADMIN.

i. Purpose. This is the subCLASS of BUILDING for category code "610", administrative building.

ii. Listing.

Listing of Class ADMIN.

```
BUILDING CLASS ADMIN;      | 610 ;  
  BEGIN FACINDX := 13; END ** ADMIN **;
```

Figure B-112

cc. Class QTRS.

i. Purpose. This is the subCLASS of BUILDING for category codes: "721", enlisted housing, "722", mess facilities, "724", officers quarters, "725", emergency housing, and "730", community facilities.

ii. Listing.

Listing of Class QTRS

```
BUILDING CLASS QTRS;      | 721 722 724 725 730 ;  
  BEGIN FACINDX := 14; END ** QTRS **;
```

Figure B-113

dd. Class PETRO.

i. Purpose. This FACILITY subCLASS is used as a prefix to CLASSES: POL, PIPELINE, and TANK. As the name implies objects from this CLASS represent facilities used for pumping, transporting and storing petroleum, oil and lubricants. The default repair method is to replace rather than fix, unless otherwise indicated.

ii. Class PETRO Skeleton.

Parameters and Attributes for Class PETRO

CLASS: PETRO	PREFIX: FACILITY
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES:	
REPAIR	PETRO FACILITYs are assumed to be destroyed when damaged unless otherwise indicated.

Figure B-114

iii. Listing.

Listing of Class PETRO.

```
FACILITY CLASS PETRO;  
  BEGIN  
  
  PROCEDURE REPAIR(EXTENT,RC,RS,LOCX,JOBQ);  
  INTEGER RC,RS,LOCX;REF(HEAD)JOBQ;REAL EXTENT;  
  BEGIN  
  PRP.ONHAND:=PRP.ONHAND - SIZE;  
  OUT;  
  END -- REPAIR --;  
  
  END ** PETRO **;
```

Figure B-115

ee. Class POLFACIL.

i. Purpose. This is the subCLASS of PETRO for the following petroleum dispensing category codes: "121", aircraft "122", marine "123", land vehicle and "126", liquid fuel.

ii. Listing.

Listing of Class POLFACIL.

```
PETRO CLASS POLFACIL;      | 121 122 123 126;  
  BEGIN FACINDX := 3; END ** POL **;
```

Figure B-116

ff. Class TANK.

i. Purpose. This is the subCLASS of PETRO for category codes "411", bulk fuel storage and "124", operational fuel storage.

ii. Listing.

Listing of Class TANK.

```
PETRO CLASS TANK;      | 411 124;
BEGIN
  PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
    REF(ORDNANCE)ORDTYP;REF(HEAD)DAMLST;REAL ORDAMT;
    BEGIN
      REAL SIZ2,DAM;REF(UNOFMS)MS;
      IF ATKLOG THEN OUTTEXT(" .TNK/DMG. ");
      MS :- CMPNT.UOM;
      IF MS /= NONE THEN
        BEGIN
          SIZ2:-IF MS.EQVMS = NOTEXT THEN SIZE ELSE SIZE*MS.COEF;
          IF ORDTYP IS BOMB THEN
            BEGIN
              IF STD = XTSTD THEN
                DAM:-BLAST("EMDA",SIZ2/HTANK,STD,ORDTYP,ORDAMT)
              ELSE
                DAM:-BLAST("MAEF",SIZ2/HBLAD,STD,ORDTYP,ORDAMT)
            END
          ELSE
            IF ORDTYP IS DEMOL THEN
              DAM :- IF STD = XISTD THEN 1.0
                ELSE IF ORDAMT > TNKDAMFAC THEN 1.0
                  ELSE 0.0;
            IF ATKLOG THEN OUTFIX(DAM,2,7);
            IF DAM > 0.0 THEN
              BEGIN
                IF DAMREP == NONE THEN
                  BEGIN
                    DAMREP:-NEW HITS(THIS TANK, 1.0);
                    DAMREP.INTO(DAMLST);
                  END;
                END;
              END;
            END;
          END -- DAMAGE --;

FACINDX := 4;
END ** TANK **;
```

Figure B-117

gg. Class PIPE.

i. Purpose. This is the subCLASS of PETRO for category code "125", pol pipeline.

ii. Listing.

Listing of Class PIPE.

```
PETRO CLASS PIPE;      | 125 ;
BEGIN

PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
  REF(ORDNANCE)ORDTYP;REF(HEAD)DAMLST;REAL ORDAMT;
  BEGIN
  REAL DAM;
  IF ATKLOG THEN OUTTEXT(" .PIPE/DMG. ");
  IF ORDTYP IS DEMOL THEN
    NEW HITS(THIS PIPE,PIPDAMFAC*ORDAMT).INTO(DAMLST)
  ELSE
    IF ORDTYP IS BOMB THEN
      BEGIN
        DAM:=BLAST(CMPNT.UOM.DAMCRIT,EMDTAB(FACINDX),
          STD,ORDTYP,ORDAMT);
        IF DAM > 0.0 THEN
          BEGIN
            NEW HITS(THIS PIPE,DAM).INTO(DAMLST);
            IF ATKLOG THEN OUTFIX(DAM,2,7);
          END;
        END;
      END;
    END -- DAMAGE --;

PROCEDURE REPAIR(EXTENT,RC,RS,LOCX,JOBQ);
  INTEGER RC,RS,LOCX;REF(HEAD)JOBQ;REAL EXTENT;
  BEGIN
  REF(POLICY)PRI;REF(COMPONENT)CT;
  PRI:=-RATE(RC,'D',LOCX,FACINDX);
  IF PRI /= NONE THEN
    BEGIN
      CT:-SELPRCMP;
      IF CKUOM(CT,EXTENT) THEN
        NEW TASK(PRI.PRIORP,RC,RS,DMGRT*CT.HORZ,DMGRT*CT.VERT,
          DMGRT*CT.OTH,CT.MINDAY,THIS FACILITY,'W').ADDTO(JOBQ);
      END;
    END -- REPAIR --;

FACINDX := 5;
END ** PIPE **;
```

Figure B-118

hh. Class UTILITY.

i. Purpose. This FACILITY subCLASS is used as a prefix to CLASSES: POWER, WATER, and WASTE. These CLASSES are characterized by their variety. Lines, generators, sewage treatment, storage tanks, etc., present a wide array of different structural types which must be handled by the damage and repair routines.

ii. Class Skeleton.

Parameters and Attributes for Class UTILITY

CLASS: UTILITY	PREFIX: FACILITY
PARAMETERS: none	
VARIABLE ATTRIBUTES:	
RPRTSK TASK	points to the repair TASK
CUMDAM real	measures cumulative damage
PROCEDURE ATTRIBUTES:	
DAMAGE	calculates utility damage
REPAIR	estimates repair workload

Figure B-119

iii. Listing.

Listing of Class UTILITY

```

FACILITY CLASS UTILITY;
BEGIN
  REAL CUMDAM;REF(TASK)RPRTSK;

  PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
    REF(ORDNANCE) ORDTYP;REAL ORDAMT;REF(HEAD)DAMLST;
  BEGIN
    REF(UNOFMS)MS;REAL DAM;

    MS :- CMPNT.UOM;
    IF ATKLOG THEN OUTTEXT(" .UTL/DMG. ");
    IF MS.EQVMS = " " THEN
      DAM:=IF MS.DAMCRIT = "EMDL" THEN
        BLAST(MS.DAMCRIT,EMDTAB(FACINDX),STD,ORDTYP,ORDAMT)
      ELSE
        BLAST(MS.DAMCRIT,SIZE,STD,ORDTYP,ORDAMT)
      ELSE
      DAM:=IF MS.DAMCRIT = "EMDL" THEN
        BLAST(MS.DAMCRIT,EMDTAB(FACINDX),STD,ORDTYP,ORDAMT)
      ELSE
        BLAST(MS.DAMCRIT,SIZE*MS.COEF,STD,ORDTYP,ORDAMT);
    IF ATKLOG THEN OUTFIX(DAM,2,7);
    IF DAM > 0 THEN
      BEGIN
        IF DAMREP == NONE THEN
          BEGIN
            DAMREP:-NEW HITS(THIS FACILITY,DAM);
            DAMREP.INTO(DAMLST);
          
```

```

        END
        ELSE
            DAMREP.EXTENT:=DAMREP.EXTENT+DAM;
        END;
    END -- DAMAGE --;

    PROCEDURE REPAIR(EXTENT,RC,RS,LOCX,JOBQ);
    INTEGER RC,RS,LOCX;REF(HEAD)JOBQ;REAL EXTENT;
    BEGIN
        REF(POLICY) PRI;REF(COMPONENT)CT;
        IF CMPNT.UOM.DAMCRIT = "EMDL" THEN
            BEGIN
                PRI:-RATE(RC,'D',LOCX,FACINDX);
                IF PRI /= NONE THEN
                    BEGIN
                        CT :- SELRPRCMP;
                        IF CKUOM(CT,EXTENT) THEN
                            NEW TASK(PRI.PRIGRP,RC,RS,DMGRT*CT.HORZ,DMGRT*CT.VERT,
                                DMGRT*CT.OTH,CT.MINDAY,THIS FACILITY,'W').ADDTO(JOBQ);
                        END;
                    END
                ELSE
                    IF BUILT THEN
                        BEGIN
                            INSPECT RPRTSK WHEN TASK DO
                                CUMDAM := CUMDAM * ((DUR-PROGRESS)/DUR) + EXTENT
                                OTHERWISE
                                    CUMDAM := CUMDAM + EXTENT;
                            PRI:-RATE(RC,'D',LOCX,FACINDX);
                            IF PRI /= NONE THEN
                                BEGIN
                                    IF CUMDAM > SIZE * SUPRFICL(FACINDX,
                                        IF STD=XISTD THEN 1 ELSE 2) THEN
                                        BEGIN
                                            IF CUMDAM > SIZE * STRUCT(FACINDX,
                                                IF STD=XISTD THEN 1 ELSE 2) THEN
                                                BEGIN
                                                    PRP.ONHAND:=PRP.ONHAND-SIZE;
                                                    IF RPRTSK /= NONE THEN RPRTSK.OUT;
                                                    OUT;
                                                    END
                                                ELSE
                                                    BEGIN
                                                        CT :- SELRPRCMP;
                                                        IF CKUOM(CT,CUMDAM) THEN
                                                            BEGIN
                                                                IF RPRTSK == NONE THEN
                                                                    BEGIN
                                                                        RPRTSK:-NEW TASK(PRI.PRIGRP,RC,RS,DMGRT*CT.HORZ,
                                                                            DMGRT*CT.VERT, DMGRT*CT.OTH,
                                                                            CT.MINDAY,THIS FACILITY,'W');
                                                                        RPRTSK.ADDTO(JOBQ);
                                                                    END
                                                                ELSE

```



```

                                RPRTSK.REVISE(DMGRT,CT);
                                END;
                                END;
                                END;
                                END;
                                END
                                ELSE
                                BEGIN
                                INSPECT WRKTSK DO PROGRESS:=ENTIER(PROGRESS*CUMDAM/SIZE);
                                CUMDAM:=0.0;
                                END;
END -- REPAIR --;

END ** UTILITY **;

```

Figure B-120

ii. Class POWER.

i. Purpose. This is the subCLASS of UTILITY for category codes "811", electric power source and "812", electric power distribution.

ii. Listing.

Listing of Class POWER

```

UTILITY CLASS POWER;      | 811 812 ;
    BEGIN FACINDX := 15; END ** POWER **;

```

Figure B-121

jj. Class WASTE.

i. Purpose. This is the subCLASS of UTILITY for category codes "831", sewage treatment/disposal and "832", sewage collection.

ii. Listing.

Listing of Class WASTE.

```

UTILITY CLASS WASTE;      | 831 832 ;
    BEGIN FACINDX := 16; END ** WASTE **;

```

Figure B-122

kk. Class WATER.

i. Purpose. This is the subCLASS of UTILITY for category codes "841", potable water supply and "842", water distribution.

ii. Listing.

Listing of Class WATER.

```
UTILITY CLASS WATER;      | 841 842 ;  
  BEGIN FACINDX :- 17; END ** WATER **;
```

Figure B-123

11. Class TRANSP0.

i. Purpose. This is the prefix CLASS to the subCLASSEs ROAD and RR. Through additions to the category codes ESC also identified tunnels and bridges and included logic in the damage routine to permit reasonable amounts of repair tasks to be generated.

ii. Class TRANSP0 Skeleton.

Parameters and Attributes for Class TRANSP0.

CLASS: TRANSP0		PREFIX: FACILITY
PARAMETERS:		
STRUC	character	'T'=tunnel, 'B'=bridge
VARIABLE ATTRIBUTES: none		

PROCEDURE ATTRIBUTES:		
DAMAGE	estimates the level of damage that occurs to the particular type of ROAD or RR object	
REPAIR	generates a new repair TASK for the object regardless of other existing repair TASKs	

Figure B-124

iii. Listing.

Listing of Class TRANSP

```
FACILITY CLASS TRANSP(STRUC);
  CHARACTER STRUC;
  BEGIN

  PROCEDURE DAMAGE(ORDTYP,ORDAMT,DAMLST);
    REF(ORDNANCE) ORDTYP;REAL ORDAMT;REF(HEAD)DAMLST;
    BEGIN | SPAN TYPES 1 ARE CANTILEVER, FILLED ARCH,
      SPAN TYPES 2 ARE SUSPEN, TRUSS, STEEL ARCH.
      (SEE WES ENGR TASK TEMPLATES FOR CORDIVEM.)
      TUNNELS ONLY RECEIVE DEBRIS CLEARING.;
    REAL DAM;
    IF ATKLOG THEN OUTTEXT(" .TRANS/DMG. ");
    IF ORDTYP IS BOMB THEN
      DAM:= IF STRUC = 'M' OR STRUC = 'S' THEN
        BLAST("EMDL",BRGFAC*EMDTAB(FACINDX),STD,ORDTYP,ORDAMT)
      ELSE IF STRUC = 'A' THEN
        BLAST("EMDL",EMDTAB(FACINDX),STD,ORDTYP,ORDAMT)
      ELSE 0.0
    ELSE
      IF ORDTYP IS DEMOL THEN
        BEGIN
          IF STRUC = 'A' THEN
            BEGIN
              IF THIS TRANSP IS RR THEN
                DAM:=ORDAMT*RRDAMFAC
              ELSE
                DAM:=ORDAMT*RDDAMFAC
            END
          ELSE
            DAM := IF STRUC = 'M' THEN SPNLEN(1)*ORDAMT/DRPSpan(1)
              ELSE IF STRUC = 'S' THEN
                SPNLEN(2)*ORDAMT/DRPSpan(2)
              ELSE IF STRUC = 'T' THEN
                ORDTYP QUA DEMOL.DEBRIS(ORDAMT)
              ELSE 0.0;
          IF DAM > 0.0 THEN
            BEGIN
              NEW HITS(THIS TRANSP,DAM);
              IF ATKLOG THEN OUTFIX(DAM,2,9);
            END;
          END;
        END;

    END -- DAMAGE --;

  PROCEDURE REPAIR(EXTENT,RC,RS,LOCK,JOBQ);
    INTEGER RC,RS,LOCK;REF(HEAD)JOBQ;REAL EXTENT;
    BEGIN
      REF(POLICY)PRI;REF(COMPONENT)CT;
      PRI:-RATE(RC,'D',LOCK,FACINDX);
      IF PRI /= NONE THEN
```

```

BEGIN
CT:-SELPRCMP;
IF CKUOM(CT,EXTENT) THEN
    NEW TASK(PRI.PRIORP,RC,RS,DMGRT*CT.HORZ,DMGRT*CT.VERT,
    DMGRT*CT.OTH,CT.MINDAY,THIS FACILITY,'W').ADDTO(JOBQ);
END;
END -- REPAIR --;
END ** TRANSP **;

```

Figure B-125

mm. Class ROAD.

- i. Purpose. This is the subCLASS of TRANSP for category code "851", roads.
- ii. Listing.

Listing of Class ROAD.

```

TRANSP CLASS ROAD;      | 851 ;
BEGIN FACINDX := 18; END ** ROAD **;

```

Figure B-126

nn. Class RR.

- i. Purpose. This is the subCLASS of RR for category code "860", railroads.
- ii. Listing.

Listing of Class RR

```

TRANSP CLASS RR;      | 860 ;
BEGIN FACINDX := 20; END ** RR **;

```

Figure B-127

oo. Class AUNIT.

- i. Purpose. This CLASS represents U.S. units that arrive, or are already deployed in the theater. The listing of AUNIT shows it to be a data structure without any operative logic (i.e., the CLASS body is empty and there is no CLASS body or procedural operatives). Each simulated day begins by reading the force deployment list (TROOP FILE). Each valid unit entry gives rise to the creation of a "NEW AUNIT". These objects may go directly to their destination, may spend some time at a reception center before moving to their destination, or, if bound for the FCZ, may go to a reception center, a staging base, and then move out of the COMMZ. Depending upon its disposition, the unit may use or require facilities at the installation it is found.

ii. Class Skeleton.

Parameters and Attributes for Class AUNIT.

CLASS: AUNIT		PREFIX: LINK
PARAMETERS:		
UTC	text	unit-type-code
SVC	character	service identifier (eg., 'A'=army)
FWD	boolean	indicates unit will probably go to FCZ
MEN	real	strength of unit from force file
TIMEOUT	real	when unit is moving through reception and staging this indicates when unit will move to next site
DEST	INSTALLATION	where the unit is going (next?)
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES: none		

Figure B-128

iii. Listing.

Listing of Class AUNIT.

```
LINK CLASS AUNIT(UTC,SVC,FWD,MEN,TIMEOUT,DEST);
  VALUE UTC;TEXT UTC; CHARACTER SVC;BOOLEAN FWD;
  REAL MEN,TIMEOUT;REF(INSTALLATION)DEST;  ;
```

Figure B-129

pp. Class REGION.

i. Purpose. One of the innovative features of SEAC is the REGION. This CLASS accomplishes two unrelated objectives: it provides a means to structure the theater in a way that accomodates both the geographic realities and engineer organizational support plans; it also is the vehicle through which runtime information is collected for post execution report generation. REGIONs are defined by the user in a tree like hierarchy of user defined depth and width. REGIONs can be broken down into subRFCIONs, which in turn can divided into other subREGIONs. In addition at the lowest subREGION in each branch of the hierarchy the MAPCELLs in the theater are assigned to particular REGIONs. The results are complementary partitions of the theater embodied in the REGION and MAPCELL definitions. Theater decomposition should follow a

rationale that supports engineer employment in the theater. The important consideration to remember (and indeed the reason why the framework was adopted) is that engineer units can be assigned to a REGION, and if so are available to work on any task that arises at INSTALLATIONS or MAPCELLs within the REGION's bounds. REGIONs have another use -- they can have special data collection object attached to them (see CLASS REPORTER). One of the problems with large detailed models like SEAC concerns the collection and presentation of information and reports. How much is enough? SEAC maintains a large amount of detailed data. If all possible information was collected each day, the model would be consumed by reporting requirements. Instead, SEAC provides the ability to capture or monitor much of what occurs, but at the user's option. There are reports to confirm that the simulation is operating correctly (e.g., switches can be set to enable input data, enemy attack, and engineer resource allocation to be monitored). But these are not the results that the user is most interested in. The "results" have to do with what facilities were required? Where are they required? When are they required? How much capability do we have and where? What work gets done? The problem is that there may be several score or hundred installations. There may also be a similar number of MAPCELLs. Then there are the many, overlapping REGIONs. To collect data and produce reports at each and every possible location is patently unrealistic. To remedy this problem ESC decided to use the flexibility of the REGION (which can represent the theater or could be defined to only represent a single MAPCELL-INSTALLATION) to focus report generation. At the beginning of an execution, REGION for which reports are wanted are identified. Data is presently collected on requirements, capabilities and utilization occurring within that REGIONs boundary. That includes all subREGIONs, all MAPCELLs, and all INSTALLATIONs that comprise the REGION.

ii. Class Skeleton.

Parameters and Attributes for Class REGION

CLASS: REGION		PREFIX: LINK
PARAMETERS:		
RX	integer	object's numeric identifier
RCOUNTRY	integer	country identifier
VARIABLE ATTRIBUTES:		
SUPRGN	REGION	REGION to which this object belongs
SUBRGNS	HEAD	set of subordinate REGIONs&MAPCELLs
ENGRASSETS	HEAD	set of engineer units in the REGION
POOL	ENGRPOOL	the engineer unit capability pool
RPTR	REPORTER	object pointer when reporting is desired
NOCANDO	boolean	flags exhaustion of this and higher REGION's engineer capability or work (saves fruitless queries)
PROCEDURE ATTRIBUTES:		
LOCGRN	REGION	searches recursively through the hierarchy for a region
CANDO	boolean	determines whether the capability in this or higher regional engineer pools can do a worktask
CAPABILITY		determines current (daily) capability of engineer units assigned to the REGION
CLCTONE		when the region has a REPORTER assignment this routine collects capability and requirements data for post execution reports
CLCTTWO		same as above but for utilization

Figure B-130

iii. Listing.

Listing of Class REGION.

```
LINK CLASS REGION(RX,RCOUNTRY);
  INTEGER RX,RCOUNTRY;
  BEGIN
    REF(REGION) SUPRGN; REF(HEAD) SUBRGNS,ENGRASSETS;
    REF(ENGRPOOL) POOL; REF(REPORTER) RPTR; BOOLEAN NOCANDO;

    REF(REGION) PROCEDURE LOCRGN(X); INTEGER X;
      BEGIN
        REF(LINK)L;REF(REGION)R;
        IF X = RX THEN LOCRGN :- THIS REGION ELSE L :- SUBRGNS.FIRST;
        WHILE L /= NONE DO
          BEGIN
            IF L IS REGION THEN
              BEGIN
                R :- L QUA REGION.LOCRGN(X);
                IF R /= NONE THEN BEGIN LOCRGN :- R; L :- NONE; END
                ELSE L:-L.SUC;
              END
            ELSE L:-L.SUC;
          END;
        END -- LOCRGN --;

    BOOLEAN PROCEDURE CANDO(RH,RV,RO,PRITY,SLICE);
      REAL RH,RV,RO;INTEGER PRITY;REAL ARRAY SLICE;
      BEGIN
        REAL HS,VS,OS;
        IF WORKLOG THEN
          BEGIN
            OUTTEXT("### CANDO REGION");OUTINT(RX,4);SETPOS(22);
            OUTFIX(RH,2,8);OUTFIX(RV,2,8);OUTFIX(RO,2,8);
            END;
          INSPECT POOL DO
            BEGIN
              IF WORKLOG THEN
                BEGIN SETPOS(50);
                  OUTFIX(HC,2,8);OUTFIX(VC,2,8);OUTFIX(OC,2,8);OUTIMAGE;
                END;
              | THE .001 TERM IS TO PREVENT NUMERIC ROUNDOFF ERRORS;
              IF HC >= (RH - .001) THEN HS:=0.0 ELSE HS:= RH - HC;
              IF VC >= (RV - .001) THEN VS:=0.0 ELSE VS:= RV - VC;
              IF OC >= (RO - .001) THEN OS:=0.0 ELSE OS:= RO - OC;
              IF HS+VS+OS = 0.0 THEN
                BEGIN
                  CANDO := TRUE;
                  HC := HC - RH;ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+RH;
                  VC := VC - RV;ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+RV;
                  OC := OC - RO;ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+RO;
                END
              ELSE
                BEGIN
                  IF OS > 0 THEN
```



```

BEGIN
IF HS = 0 THEN
  BEGIN
  IF VS = 0 THEN
    BEGIN INTEGER XHV;
    XHV := (HC - RH) + (VC - RV);
    IF XHV > OS THEN
      BEGIN
        RH:=RH+OS*((HC-RH)/XHV);
        RV:=RV+OS*((VC-RV)/XHV);
        OS := 0; RO := OC;
      END
    ELSE
      BEGIN
        RH:=HC;RV:=VC;OS:=OS-XHV;
      END;
    END
  ELSE
    IF HC > (RH + OS) THEN
      BEGIN RH:=RH+OS;RO:=OC;OS:=0;END
    ELSE BEGIN OS:=OS-(HC-RH);RH:=HC;END;
  END
  ELSE
    IF VS = 0 THEN
      BEGIN
        IF VC > (RV + OS) THEN
          BEGIN RV:=RV+OS;RO:=OC;OS:=0;END
        ELSE BEGIN OS:=OS-(VC-RV);RV:=VC;END;
        END;
      END;
    END;
  IF HS+VS+OS = 0.0 THEN
    BEGIN
    CANDO := TRUE;
    HC := HC - RH;ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+RH;
    VC := VC - RV;ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+RV;
    OC := OC - RC;ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+RO;
    END
    ELSE
    IF SUPRGN == NONE THEN
      BEGIN
      CANDO := FALSE;
      IF HC+VC+OC <= 0.0 THEN NOCANDO := TRUE;
      SLICE(1):=HS;SLICE(2):=VS;SLICE(3):=OS;
      END
    ELSE
      IF SUPRGN.NOCANDO THEN
        BEGIN
        CANDO := FALSE;
        IF HC+VC+OC <= 0.0 THEN NOCANDO := TRUE;
        SLICE(1):=HS;SLICE(2):=VS;SLICE(3):=OS;
        END
      ELSE
        IF SUPRGN.CANDO(HS,VS,OS,PRITY,SLICE) THEN
          BEGIN

```

```

        CANDO := TRUE;
        IF HS = 0 THEN
            BEGIN HC:=HC - RH;
            ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+RH; END
            ELSE
            BEGIN
            ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+HC;
            HC := 0.0;
            END;
        IF VS = 0 THEN
            BEGIN VC:=VC - RV;
            ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+RV; END
            ELSE
            BEGIN
            ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+VC;
            VC := 0.0;
            END;
        IF OS = 0 THEN
            BEGIN OC:=OC - RO;
            ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+RO; END
            ELSE
            BEGIN
            ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+OC;
            OC := 0.0;
            END;
        END
        ELSE
        CANDO := FALSE;
    END;
END
    OTHERWISE
    BEGIN
    IF WORKLOG THEN OUTIMAGE;
    IF SUPRCN /= NONE THEN
        CANDO := SUPRCN.CANDO(RH,RV,RO,PRITY,SLICE)
        ELSE
        BEGIN
        CANDO := FALSE; SLICE(1):=HS; SLICE(2):=VS; SLICE(3):=OS;
        END;
    END;
END -- CANDO --;

PROCEDURE CAPABILITY(LPOS,REP); BOOLEAN REP; INTEGER LPOS;
    BEGIN
    REF(LINK)R; REF(GEOLOC)G; REF(INSTALLATION)I;

    PROCEDURE SHOW(P); REF(ENGRPOOL)P;
        BEGIN
        IF REP THEN
            BEGIN SETPOS(53);OUTFIX(P.HC,1,8);
            OUTFIX(P.VC,1,8);OUTFIX(P.OC,1,8);OUTIMAGE;
            END;
        END -- SHOW --;

```

```

IF REP THEN
  BEGIN SETPOS(LPOS);
  OUTTEXT("
  SETPOS(LPOS);OUTINT(RX,5);
  END;
NOCANDO := FALSE;
IF POOL /= NONE THEN
  BEGIN POOL.TALLY(ENGRASSETS,REP); SHOW(POOL); END
  ELSE
  IF NOT ENGRASSETS.EMPTY THEN
    BEGIN POOL:-NEW ENGRPOOL;
    POOL.TALLY(ENGRASSETS,REP);
    SHOW(POOL);
    END;
IF POOL == NONE THEN
  BEGIN IF SUPRGN == NONE THEN NOCANDO := TRUE
  ELSE IF SUPRGN.NOCANDO THEN NOCANDO := TRUE;
  END;
R:-SUBRGNS.FIRST;
WHILE R/=NONE DO
  BEGIN
  INSPECT R
    WHEN REGION DO CAPABILITY(LPOS+5,REP)
    WHEN REGCELL DO
      INSPECT TC WHEN MAPCELL DO
        BEGIN
        IF REP THEN
          BEGIN SETPOS(40);OUTTEXT("AREA ");END;
        G:-GEOSSET.FIRST;
        WHILE G/=NONE DO
          BEGIN
          IF REP THEN
            BEGIN SETPOS(47);OUTTEXT(G.GEOLAB);END;
          I:-G.SITES.FIRST;
          WHILE I/=NONE DO
            BEGIN
            INSPECT I DO IF COUN = RCOUNTRY THEN
              BEGIN
              CANTDO := FALSE;
              IF POOL/=NONE THEN
                BEGIN
                POOL.TALLY(ENGRS,REP);
                SHOW(POOL);
                END
              ELSE
                IF NOT ENGRS.EMPTY THEN
                  BEGIN POOL:-NEW ENGRPOOL;
                  POOL.TALLY(ENGRS,REP);
                  SHOW(POOL);
                  END
                ELSE
                  IF NOCANDO THEN
                    CANTDO := TRUE;
              END;
            END;
          END;
        END;
  END;

```

```

                                I :- I.SUC;
                                END;
                                G :- G.SUC;
                                END;
                                END;

                                R:-R.SUC;
                                END;
                                END -- CAPABILITY --;

PROCEDURE CLCTONE(REF);REF(REPORTER)REP;
BEGIN REF(LINK)R;REF(GEoloc)G;REF(INSTALLATION)I;INTEGER J;

PROCEDURE EDATA(C,S,P);REAL ARRAY C,S;REF(ENGRPOOL)P;
IF P /= NONE THEN
BEGIN
C(1):=C(1) + P.HC;
C(2):=C(2) + P.VC;
C(3):=C(3) + P.OC;
FOR J:=1,2,3,4,5 DO S(J) := S(J) + P.ESVC(J);
END -- EDATA --;

PROCEDURE TDATA(A,B,TQ);REAL ARRAY A,B;REF(HEAD) TQ;
BEGIN REF(TASK) T;
T :- TQ.FIRST;
WHILE T /= NONE DO
BEGIN
IF ROUNTRY = T.NAT THEN
INSPECT T DO
BEGIN
A(PRIO,1):=A(PRIO,1)+WHRZ;
A(PRIO,2):=A(PRIO,2)+WVER;
A(PRIO,3):=A(PRIO,3)+WOTH;
IF TIME <= TIMIN+DUR-1 THEN
BEGIN
IF WRKTYP = 'C' THEN
B(SVC,1):=B(SVC,1)+DAILY
ELSE
IF WRKTYP = 'W' THEN
B(SVC,2):=B(SVC,2)+DAILY;
END;
END;
T :- T.SUC;
END;
END --- TDATA ---;

EDATA(REF.CAP,REF.SVCAP,POOL);
R :- SUBRGNS.FIRST;
WHILE R /= NONE DO
BEGIN
INSPECT R WHEN REGION DO
BEGIN
IF RPTR /= NONE THEN
BEGIN
RPTR.ZED;

```

```

        CLCTONE(RPTR);
        FOR J := 1,2,3,4 DO
            BEGIN
                REP.RQT(J,1):=REP.RQT(J,1) + RPTR.RQT(J,1);
                REP.RQT(J,2):=REP.RQT(J,2) + RPTR.RQT(J,2);
                REP.RQT(J,3):=REP.RQT(J,3) + RPTR.RQT(J,3);
            END;
        FOR J := 1,2,3,4,5 DO
            BEGIN
                REP.WRKCAT(J,1):=REP.WRKCAT(J,1)+RPTR.WRKCAT(J,1);
                REP.WRKCAT(J,2):=REP.WRKCAT(J,2)+RPTR.WRKCAT(J,2);
                REP.SVCAP(J) := REP.SVCAP(J) + RPTR.SVCAP(J);
            END;
        REP.CAP(1):=REP.CAP(1)+RPTR.CAP(1);
        REP.CAP(2):=REP.CAP(2)+RPTR.CAP(2);
        REP.CAP(3):=REP.CAP(3)+RPTR.CAP(3);
        END
        ELSE
            CLCTONE(REP);
        END
    WHEN REGCELL DO
        BEGIN
            IF TC IS MAPCELL THEN
                BEGIN
                    TDATA(REP.RQT,REP.WRKCAT,TC.TASKLIST);
                    G :- TC.GEASET.FIRST;
                    WHILE G /= NONE DO
                        BEGIN
                            I:-G.SITES.FIRST;
                            WHILE I /= NONE DO
                                BEGIN
                                    IF I.COUN = RCOUNTRY THEN
                                        BEGIN
                                            TDATA(REP.RQT,REP.WRKCAT,I.JOBS);
                                            EDATA(REP.CAP,REP.SVCAP,I.POOL);
                                        END;
                                    I :- I.SUC;
                                END;
                            G :- G.SUC;
                        END;
                    END;
                END;
            R :- R.SUC;
        END;
        IF REP == RPTR THEN REP.ONEOUT;
        END -- CLCTONE ---;

PROCEDURE CLCTTWO(REP);REF(REPORTER)REP;
    BEGIN REF(LINK)R;REF(GEOLOC)G;REF(INSTALLATION)I;

PROCEDURE UDATA(U,P);REAL ARRAY U;REF(ENGRPOOL)P;
    BEGIN INTEGER I;
    IF P /= NONE THEN
        FOR I := 1, 2, 3, 4 DO

```

```

        BEGIN
        U(I,1):=U(I,1) + P.ENGRUSE(I,1);
        U(I,2):=U(I,2) + P.ENGRUSE(I,2);
        U(I,3):=U(I,3) + P.ENGRUSE(I,3);
        END;
    END -- UDATA --;

    UDATA(REP.UTLZ,POOL);
    R :- SUBRGNS.FIRST;
    WHILE R /= NONE DO
        BEGIN
            INSPECT R WHEN REGION DO
                BEGIN
                    IF RPTR /= NONE THEN
                        BEGIN INTEGER P;
                        CLCTTWO(RPTR);
                        FOR P := 1,2,3,4 DO
                            BEGIN
                                REP.UTLZ(P,1):=REP.UTLZ(P,1) + RPTR.UTLZ(P,1);
                                REP.UTLZ(P,2):=REP.UTLZ(P,2) + RPTR.UTLZ(P,2);
                                REP.UTLZ(P,3):=REP.UTLZ(P,3) + RPTR.UTLZ(P,3);
                            END;
                        END
                    ELSE
                        CLCTTWO(REP);
                    END
                WHEN REGCELL DO
                    BEGIN
                        IF TC IS MAPCELL THEN
                            BEGIN
                                G :- TC.GEASET.FIRST;
                                WHILE G /= NONE DO
                                    BEGIN
                                        I:-G.SITES.FIRST;
                                        WHILE I /= NONE DO
                                            BEGIN
                                                IF I.COUN = RCOUNTRY THEN
                                                    UDATA(REP.UTLZ,I.POOL);
                                                I :- I.SUC;
                                            END;
                                        G :- G.SUC;
                                    END;
                                END;
                            END;
                        END;
                    END;
                R :- R.SUC;
            END;
        IF REP == RPTR THEN REP.TWOUT;
        END -- CLCTTWO --;

    SUBRGNS :- NEW HEAD;
    ENGRASSETS :- NEW HEAD;
    END *** REGION *** ;

```

Figure B-131

qq. Class REGCELL.

i. Purpose. This CLASS is used to put MAPCELL pointers in a REGION's subordinate list. Since MAPCELL is not prefixed by LINKAGE, it cannot be INCLUDED in the set established by SUBRGNS. Since there may be more than one regional hierarchy (in a multi-national COMMZ), prefixing MAPCELL with LINK would not solve the problem because it could then only belong to one REGION object. REGCELL was defined to bridge this software gap. REGCELL objects are created for each REGION-MAPCELL relation.

ii. Class Skeleton.

Parameters and Attributes for Class REGCELL

CLASS: REGCELL		PREFIX: LINK	
.PARAMETERS:			
TC	MAPCELL	indicates a mapcell that is de-	
		fined to be in the REGION	
VARIABLE ATTRIBUTES: none			
PROCEDURE ATTRIBUTES: none			

Figure B-132

iii. Listing.

Listing of Class REGCELL.

```
LINK CLASS REGCELL(TC); REF(MAPCELL)TC; ;
```

Figure B-133

rr. Class HITS.

i. Purpose. SEAC explicitly carries out attacks by enemy planes and ground troops against facilities. Since a facility may be hit by more than one attack it proved necessary to accumulate the damage from successive attacks. When an attack results in damage a new HITS object is created and placed in the DAMLIST set. All subsequent attacks carried out will modify the extent of damage that is contained in the HITS object associated with the particular FACILITY being attacked.

ii. Class Skeleton.

Parameters and Attributes for Class HITS

CLASS: HITS		PREFIX: LINK
PARAMETERS:		
FAC	FACILITY	points to the object that received damage
EXTENT	real	the amount of calculated damage
VARIABLE ATTRIBUTES: none		
PROCEDURE ATTRIBUTES: none		

Figure B-134

iii. Listing.

Listing of Class HITS

```
LINK CLASS HITS(FAC,EXTENT); REF(FACILITY)FAC;REAL EXTENT;  
BEGIN  
  IF ATKLOG THEN  
    BEGIN OUTTEXT("&HIT& DAMAGE OF ");  
    IF FAC.PRP -/- NONE THEN OUTTEXT(FAC.PRP.CTCODE)  
      ELSE BEGIN OUTTEXT(" MSR#"); OUTINT(FAC.FACINDX,3);END;  
    OUTFIX(EXTENT,1,7);OUTTEXT(" AT TIME");  
    OUTFIX(TIME,2,6);OUTIMAGE;  
    END;  
  END *** HITS ***;
```

Figure B-135

ss. Class TASK.

i. Purpose. Engineer work projects are represented at INSTALLATIONS and MAPCELLS by objects from CLASS TASK. The type of work (construction or repair), its priority (vital, critical, essential, or necessary), the duration, the average daily hours by skill (horizontal, vertical, and other), and the requesting country and service are all reflected in the job data. Each day the available engineer capability is apportioned to projects to accomplish the highest priority TASKs, to the extent of available assets. When TASKs are finished they are removed from the job queues and the FACILITY is either constructed or repaired. (Presently no special consideration is given to projects that stay in the queue for an inordinate length of time. Because of this at the end of the simulation we find some TASKs that have been in the queue since day 0 but have not been performed because the priority was low, or there was insufficient capability. There probably should be a means to either drop or elevate these projects.)

ii. Class Skeleton.

Parameters and Attributes for Class TASK

CLASS: TASK		PREFIX: LINK
PARAMETERS:		
PRIO	integer	numeric priority (eg., 1=vital)
NAT	integer	nationality
SVC	integer	service that generated TASK
WHRZ	real	daily horizontal requirements
WVER	real	" vertical "
WOTH	real	" other "
DUR	real	minimum days to complete work
FACPTR	FACILITY	the FACILITY that caused this TASK
WRKTYP	character	w=damage, c=construction
VARIABLE ATTRIBUTES:		
PROGRESS	real	days of work accomplished
DAILY	real	daily horizontal+vertical+other hrs
TIMIN	real	day the TASK was generated
PROCEDURE ATTRIBUTES:		
DUMP		prints objects's content
PERFORM		changes attributes when work is performed
REVISE		adjusts damage when additional hits occur
ADDTO		places object in job queue according to priority and amount of work required

Figure B-136

iii. Listing.

Listing of Class TASK

```

LINK CLASS TASK(PRIO,NAT,SVC,WHRZ,WVER,WOTH,DUR,FACPTR,WRKTYP);
INTEGER PRIO,NAT,SVC;REAL WHRZ,WVER,WOTH,DUR;
REF(FACILITY)FACPTR;CHARACTER WRKTYP;
BEGIN
REAL PROGRESS,DAILY,TIMIN;

PROCEDURE DUMP;
BEGIN SETPOS(40);OUTFIX(TIMIN,1,5);
OUTFIX(PROGRESS,1,6);OUTCHAR('/');OUTFIX(DUR,1,4);
OUTFIX(DAILY,2,8);OUTINT(PRIO,4);OUTCHAR('-');
IF FACPTR /= NONE THEN

```

```

        BEGIN
        OUTTEXT(FACPTR.PRP.CTCODE);OUTCHAR('/');
        OUTFIX(FACPTR.SIZE,2,12);
        END
        ELSE OUTTEXT("MSR/");
        OUTCHAR(WRKTP);
        OUTIMAGE;
        END;

PROCEDURE PERFORM(PORZION);REAL PORZION;
BEGIN
    PROGRESS := PROGRESS + PORZION;
    IF PROGRESS >= DUR THEN
        BEGIN
            OUT;
            IF NOT FACPTR.BUILT THEN FACPTR.FINISHED
            ELSE
                IF FACPTR.WRKTSK == THIS TASK THEN
                    FACPTR.WRKTSK :- NONE;
                END
            ELSE
                IF DUR-PROGRESS < 1 THEN
                    BEGIN
                        PROGRESS := DUR - PROGRESS;
                        WHRZ := PROGRESS * WHRZ;
                        WVER := PROGRESS * WVER;
                        WOTH := PROGRESS * WOTH;
                        DUR := 1.0;PROGRESS := 0.0;
                    END;
                END-- PERFORM --;
            END
        END

PROCEDURE ADDTO(Q); REF(HEAD)Q;
BEGIN
    REF(TASK)T;
    T :- Q.FIRST;
    IF T == NONE THEN INTO(Q)
    ELSE
        WHILE T /= NONE DO
            IF PRIO > T.PRIO THEN T :- T.SUC
            ELSE
                IF PRIO < T.PRIO THEN
                    BEGIN PRECEDE(T); T :- NONE; END
                ELSE
                    IF DAILY <= T.DAILY THEN
                        BEGIN PRECEDE(T); T :- NONE; END
                    ELSE T :- T.SUC;
                END
            END
        END
    IF PREV == NONE THEN INTO(Q);
    END -- ADD_TO_JOBS --;

PROCEDURE REVISE(MULT,RPRCT);REAL MULT;REF(COMPONENT)RPRCT;
BEGIN
    WHRZ:=MULT*RPRCT.HORZ;
    WVER:=MULT*RPRCT.VERT;
    WOTH:=MULT*RPRCT.OTH;

```

```

    PROGRESS := 0.0; TIMIN := TIME;
    DAILY := WHRZ+WVER+WOTH;
    END -- REVISE --;

TIMIN := TIME;DAILY := WHRZ+WVER+WOTH;
END ** TASK **;

```

Figure B-137

tt. Class INSTALLATION.

i. Purpose. Objects from this CLASS play a major role in SEAC. Most requirements generated by SEAC result from unit needs. Units arrive at and move to INSTALLATIONS. Facility requirements are generated at INSTALLATIONS. This is the first step in a series of actions that are carried out each day at every installation included in the simulation: arriving and transient units are processed, new facility requirements are calculated, damage is assessed, new engineer tasks are placed into a job queue, engineer unit capability is determined, projects are selected from the job queue for work according to priorities, and upon completion jobs are removed from the queue and facility and real property information maintained in the installation updated accordingly. As one can see much can occur within an installation. INSTALLATION also has several subCLASSES (eg., AIRBASEs, PORTs, DEPOTs, etc.) that permit additional data to be maintained and special processing rules to be implemented (primarily through SIMULA's VIRTUAL procedure concept).

ii. Class INSTALLATION Skeleton.

Parameters and Attributes for Class INSTALLATION

CLASS: INSTALLATION		PREFIX: LINK
PARAMETERS:		
COUN	integer	country index
SRVC	integer	service index
INAME	text	name of installation
VIRTUAL:		
CHECKIN		processes arriving units
STATUS		produces installation report
TGTSELECT		randomly selects a target facility from facility group
GETS		adds facility to appropriate group
VARIABLE ATTRIBUTES:		
JOBS	HEAD	contains INSTALLATION's TASKs
ENGRS	HEAD	contains INSTALLATION's engineer units
ARRIVALS	HEAD	units arriving at the INSTALLATION that day
FBLDG	HEAD	building+ assets
FPOL	HEAD	POL facilities
FUTIL	HEAD	utility assets

FROAD	HEAD	roadway assets
FRUN	HEAD	runway assets
DAMLIST	HEAD	list of HITS received that day
POOL	ENGRPOOL	object to accumulate capability
JOB	TASK	used to refer to individual TASKs in JOBS
ASSETS	REALTYTREE	AVL tree of asset tallies
SORTIESRECD	integer	air sorties count
RCATK	integer	ground attack count
BOMBHITS	integer	bombs dropped
ROCHITS	integer	rockets fired at
INDX	integer	INSTALLATION category index
CANTDO	boolean	flag used when capability exhausted
POPULATION	real	permanent party population

PROCEDURE ATTRIBUTES:		
DAMCNTRL		examines the HITS entries (if any) and initiates repair determination
CHECKIN		takes units (see AUNIT) from arrival queue and in-processes then
DETRQMT		the FACILITY requirements for each arriving unit are determined
GETS		places a new FACILITY into its proper category
TGTSELECT	FACILITY	randomly selects a target from a FACILITY set based on the aimpoint of the attacking plane/unit
STATUS		prints INSTALLATION data
TRIAGE		goes through the task list and assigns jobs according to priority and available engineer capability
CANDO	boolean	this procedure determines when a job can be done based on capability at the INSTALLATION and in area support to it (i.e., capability at the REGIONS)

Figure B-138

iii. Listing.

Listing of Class INSTALLATION

```
LINK CLASS INSTALLATION(COUN,SRVC,INAME);
VALUE INAME;TEXT INAME;
INTEGER COUN,SRVC;
VIRTUAL : PROCEDURE CHECKIN,STATUS,TGTSELECT,GETS;
BEGIN
REF(HEAD) JOBS,ENGRS,ARRIVALS,FBLDG,FPOL,FUTIL,FROAD,FRUN,DAMLIST;
REF(HEAD) ARRAY FACLS(1:5);
REF(ENGRPOOL)POOL;      REF(TASK)JOB; REF(REALTYTREE)ASSETS;
INTEGER SORTIESRECVD, RCATK, BOMBHITS, ROCHITS, INDX;
BOOLEAN CANTDO; REAL POPULATION;

PROCEDURE DAMCNTRL;
BEGIN
REF(HITS)H;
FOR H:-DAMLIST.FIRST WHILE H /= NONE DO
BEGIN
H.FAC.REPAIR(H.EXTENT,COUN,SRVC,INDX,JOBS);
IF H.FAC.DAMREP == H THEN H.FAC.DAMREP :- NONE;
H.OUT;
END;
END --DAMCNTRL--;

PROCEDURE CHECKIN;
BEGIN
REF(AUNIT) A;REAL NEWTROOPS;
FOR A:-ARRIVALS.FIRST WHILE A /= NONE DO
BEGIN
NEWTROOPS:=NEWTROOPS + A.MEN;
DETRQMT(A.UTC,1.0);
A.OUT;
END;
DETRQMT("PEOPLE",NEWTROOPS);
POPULATION := POPULATION + NEWTROOPS;
END -- CHECKIN --;

PROCEDURE DETRQMT(KEY,MAG); TEXT KEY;REAL MAG;
BEGIN
REF(FACELEM) ELT; REF(FACTOR) F; REF(PROPERTY) RP;
IF MAG > 0.0 THEN
BEGIN
F :- FACTAB.FIND(KEY);
IF F IS FACTORX THEN F :- F QUA FACTORX.SEL(COUN,SRVC);
IF F /= NONE THEN ELT:-F.FIRST;
WHILE ELT /= NONE DO
BEGIN
IF ELT.UOM = "XX" THEN
DETRQMT(ELT.CODE,ELT.RATE)
ELSE
BEGIN
```

```

        RP :-ASSETS.FIND(ELT.CODE);
        IF RP == NONE THEN
            BEGIN
                RP :- NEW PROPERTY(ELT.CODE);
                IF RP.XREF(THIS INSTALLATION) THEN RP.RANK(ASSETS)
                    ELSE RP :- NONE;
            END;
        IF RP /= NONE THEN RP.NEEDS(ELT.RATE*MAG,ELT.UOM);
        END;
        ELT :- ELT.NX;
        END;
    END;
END -- DETRQMT --;

PROCEDURE GETS(F,C);REF(FACILITY)F;TEXT C;
    IF F IN BUILDING THEN F.INTO(FBLDG)
    ELSE IF F IN PETRO THEN F.INTO(FPOL)
    ELSE IF F IN UTILITY THEN F.INTO(FUTIL)
    ELSE IF F IN TRANSPOR THEN F.INTO(FROAD)
    ELSE IF F IN SURFACE THEN F.INTO(FRUN);

REF(FACILITY) PROCEDURE TGTSELECT(K); INTEGER K;
    BEGIN
        IF K = 0 THEN K := DISCRETE(TGTDEFAULTS(INDX).VECTOR,UTSEL);
        IF FACLIS(K).EMPTY THEN
            BEGIN INTEGER L;
                FOR L := 1 STEP 1 UNTIL 3 DO
                    BEGIN
                        K := DISCRETE(TGTDEFAULTS(INDX).VECTOR,UTSEL);
                        IF NOT FACLIS(K).EMPTY THEN L := 3;
                    END;
                IF K = 0 THEN TGTSELECT :- NONE
                    ELSE TGTSELECT :- RANDPICK(FACLIS(K));
            END
        ELSE
            TGTSELECT :- RANDPICK(FACLIS(K));
        END -- TGTSELECT --;

PROCEDURE STATUS(ASTA,ESTA,TSTA);BOOLEAN ASTA,ESTA,TSTA;
    BEGIN REF(TASK) TSK;
        OUTTEXT(" ---[ ");OUTTEXT(INAME);
        OUTCHAR('/');OUTINT(COUN,1);OUTCHAR('/');OUTINT(SRVC,1);OUTCHAR('/');
        OUTTEXT(" ] STATUS REPORT.");SETPOS(50);OUTTEXT(" POPULATION:");
        OUTFIX(POPULATION,0,7); OUTTEXT(" SORTIES:");OUTINT(SORTIESRECD,5);
        OUTTEXT(" SPF ATTACKS:");OUTINT(RCATK,5); OUTIMAGE;
        IF ASTA THEN ASSETS.TRAVERSE;
        OUTTEXT(" ENGR UNITS");OUTINT(ENGRS.CARDINAL,6);
        IF ESTA AND POOL /= NONE THEN POOL.TALLY(ENGRS,TRUE);
        OUTTEXT(" TASKS");OUTINT(JOBS.CARDINAL,6);
        IF TSTA THEN
            BEGIN
                TSK :- JOBS.FIRST;
                IF TSK == NONE THEN OUTIMAGE
            ELSE

```

```

        WHILE TSK /= NONE DO BEGIN TSK.DUMP; TSK := TSK.SUC; END;
    END
    ELSE
        OUTIMAGE;
    END -- STATUS --;

PROCEDURE TRIAGE(MPCL,PRIX); REF(MAPCELL)MPCL; INTEGER PRIX;
BEGIN
    REF(TASK)T;REF(LINK)JT;REAL ARRAY PIECE(1:3);REAL MINPIECE;

    IF PRIX = 1 THEN T := JOB := JOBS.FIRST ELSE T := JOB;
    WHILE T /= NONE DO
        IF CANTDO THEN T:= NONE
        ELSE
            IF JOB.PRIO <= PRIX THEN
                BEGIN
                    IF CANDO(MPCL,JOB.WHRZ,JOB.WVER,JOB.WOTH,PRIX,PIECE) THEN
                        BEGIN
                            T:=-JOB.SUC;
                            JOB.PERFORM(1.0);
                            JOB := T;
                            END
                        ELSE
                            BEGIN
                                PIECE(1):=-IF JOB.WHRZ = 0 THEN 0.0
                                    ELSE PIECE(1)/JOB.WHRZ;
                                PIECE(2):=-IF JOB.WVER = 0 THEN 0.0
                                    ELSE PIECE(2)/JOB.WVER;
                                PIECE(3):=-IF JOB.WOTH = 0 THEN 0.0
                                    ELSE PIECE(3)/JOB.WOTH;
                                IF PIECE(1) > PIECE(2) THEN
                                    BEGIN
                                        IF PIECE(1) > PIECE(3) THEN MINPIECE:=-1-PIECE(1)
                                        ELSE MINPIECE := 1 - PIECE(3);
                                    END
                                ELSE
                                    IF PIECE(2) > PIECE(3) THEN MINPIECE:=-1-PIECE(2)
                                    ELSE MINPIECE := 1 - PIECE(3);
                                IF MINPIECE >= PIECEWORK THEN
                                    BEGIN IF CANDO(MPCL,MINPIECE*JOB.WHRZ,
                                        MINPIECE*JOB.WVER, MINPIECE*JOB.WOTH,
                                        PRIX,PIECE) THEN
                                        BEGIN
                                            T:=-JOB.SUC;
                                            JOB.PERFORM(MINPIECE);
                                            JOB := T;
                                            END
                                        ELSE
                                            T := JOB := JOB.SUC;
                                        END
                                    ELSE
                                        T := JOB := JOB.SUC;
                                    END
                                ELSE
                                    T := JOB := JOB.SUC;
                                END
                            END
                        END
                    END
                END
            END
        END
    END
END

```



```

        ELSE
        T :- NONE;
END -- TRIAGE --;

BOOLEAN PROCEDURE CANDO(MC,RH,RV,RO,PRITY,SLICE);
REF(MAPCELL)MC;REAL RH,RV,RO;INTEGER PRITY; REAL ARRAY SLICE;
BEGIN
REAL HS,VS,OS;
REF(REGION)RGN;
IF WORKLOG THEN
    BEGIN OUTTEXT("### ");OUTTEXT(INAME);SETPOS(22);
    OUTFIX(RH,2,8);OUTFIX(RV,2,8);OUTFIX(RO,2,8);
    END;
INSPECT POOL DO
    BEGIN
    IF WORKLOG THEN
        BEGIN SETPOS(50);OUTFIX(HC,2,8);OUTFIX(VC,2,8);
        OUTFIX(OC,2,8);OUTIMAGE;
        END;
    | THE .001 TERM IS TO PREVENT NUMERIC ROUNDOFF ERRORS;
    IF HC >= (RH - .001) THEN HS:=-0.0 ELSE HS:=- RH - HC;
    IF VC >= (RV - .001) THEN VS:=-0.0 ELSE VS:=- RV - VC;
    IF OC >= (RO - .001) THEN OS:=-0.0 ELSE OS:=- RO - OC;
    IF HS+VS+OS = 0.0 THEN
        BEGIN
        CANDO := TRUE;
        HC := HC - RH;ENGRUSE(PRITY,1):=-ENGRUSE(PRITY,1)+RH;
        VC := VC - RV;ENGRUSE(PRITY,2):=-ENGRUSE(PRITY,2)+RV;
        OC := OC - RO;ENGRUSE(PRITY,3):=-ENGRUSE(PRITY,3)+RO;
        END
        ELSE
        BEGIN
        IF OS > 0 THEN
            BEGIN
            IF HS = 0 THEN
                BEGIN
                IF VS = 0 THEN
                    BEGIN INTEGER XHV;
                    XHV := (HC - RH) + (VC - RV);
                    IF XHV >= OS THEN
                        BEGIN
                        RH:=-RH+OS*((HC-RH)/XHV);
                        RV:=-RV+OS*((VC-RV)/XHV);
                        OS := 0; RO := OC;
                        END
                        ELSE
                        BEGIN
                        RH:=-HC;RV:=-VC;OS:=-OS-XHV;
                        END;
                    END
                ELSE
                BEGIN
                IF HC > (RH + OS) THEN
                    BEGIN RH:=-RH+OS;RO:=-OC;OS:=0;END
                    ELSE BEGIN OS:=-OS-(HC-RH);RH:=-HC;END;
                END
            END
        END
    END

```

```

END
ELSE
IF VS = 0 THEN
BEGIN
IF VC > (RV + OS) THEN
BEGIN RV:=RV+OS;RO:=OC;OS:=0;END
ELSE BEGIN OS:=OS-(VC-RV);RV:=VC;FND;
END;
END;
IF HS+VS+OS = 0.0 THEN
BEGIN
CANDO := TRUE;
HC := HC - RH; ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+RH;
VC := VC - RV; ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+RV;
OC := OC - RO; ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+RO;
END
ELSE
BEGIN
RGN :- MC.RGNLINK(COUN);
IF RGN.NOCANDO THEN
BEGIN
IF HC+VC+OC <= 0.0 THEN CANTDO:=TRUE;
CANDO:=FALSE;
SLICE(1):=HS;SLICE(2):=VS;SLICE(3):=OS;
END
ELSE
IF RGN.CANDO(HS,VS,OS,PRITY,SLICE) THEN
BEGIN
CANDO := TRUE;
IF HS = 0 THEN
BEGIN HC:=HC - RH;
ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+RH;END
ELSE
BEGIN
ENGRUSE(PRITY,1):=ENGRUSE(PRITY,1)+HC;
HC := 0.0;
END;
IF VS = 0 THEN
BEGIN VC:=VC - RV;
ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+RV;END
ELSE
BEGIN
ENGRUSE(PRITY,2):=ENGRUSE(PRITY,2)+VC;
VC := 0.0;
END;
IF OS = 0 THEN
BEGIN OC:=OC - RO;
ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+RO;END
ELSE
BEGIN
ENGRUSE(PRITY,3):=ENGRUSE(PRITY,3)+OC;
OC := 0.0;
END;
END
END
END

```

```

                                ELSE
                                CANDO := FALSE;
                                END;
                                END;
                                END
                                OTHERWISE
                                BEGIN
                                IF WORKLOG THEN OUTIMAGE;
                                RGN :- MC.RGNLINK(COUN);
                                IF RGN /= NONE THEN
                                    CANDO := RGN.CANDO(RH,RV,RO,PRITY,SLICE)
                                    ELSE
                                    BEGIN
                                    CANDO:=FALSE;SLICE(1):=-HS;SLICE(2):=-VS;SLICE(3):=-OS;
                                    END;
                                END;
                                END;
                                END -- CANDO --;

                                JOBS :- NEW HEAD;  ENGRS :- NEW HEAD;
                                ARRIVALS :- NEW HEAD;  DAMLIST :- NEW HEAD;
                                FACLIS(1) :- FRUN :- NEW HEAD;
                                FACLIS(2) :- FBLDG :- NEW HEAD;
                                FACLIS(3) :- FPOL :- NEW HEAD;
                                FACLIS(4) :- FROAD :- NEW HEAD;
                                FACLIS(5) :- FUTIL :- NEW HEAD;
                                INDX :- 9;
                                ASSETS :- NEW REALTYTREE("INSTALLATION ASSETS",
                                                            "      REQD      ONHAND UNDERWAY");
                                END ** INSTALLATION ** ;

```

Figure B-139

uu. Class AIRBASE.

i. Purpose. This subCLASS of INSTALLATION expands the number of facility categories to better represent the results and intentions of enemy attacks against an AIRBASE. A parameter is also added to indicate those airbases that are collocated (this designation will effect which country/service has possession and therefore responsibility for major surface and operational FACILITIES.

ii. Class AIRBASE Skeleton.

Parameters and Attributes for Class AIRBASE

CLASS: AIRBASE		PREFIX: INSTALLATION
PARAMETERS: none		
VARIABLE ATTRIBUTES:		
FRAMES	HEAD	not used
COLOC	AIRBASE	pointer to a collocated AIRBASE
PROCEDURE ATTRIBUTES:		
TGTSELECT	similar to procedure defined in prefix CLASS but uses more facility categories.	
GETS	assigns new FACILITYs to the expanded category set array	

Figure B-140

iii. Listing.

Listing of Class AIRBASE

```

INSTALLATION CLASS AIRBASE;
BEGIN
  REF(HEAD) ARRAY FACLISX(6:9);
  REF(HEAD)FRAMES; REF(AIRBASE)COLOC;

  REF(FACILITY) PROCEDURE TGTSELECT(K); INTEGER K;
  NEEDS REVISION
  BEGIN
    IF K = 0 THEN K := DISCRETE(TGTDEFAULTS(INDX).VECTOR,UTSEL);

    IF (IF K LE 5 THEN FACLIS(K).EMPTY ELSE FACLISX(K).EMPTY) THEN
      BEGIN
        INTEGER ALTGT;
        ALTGT := 3;
        WHILE ALTGT > 0 DO
          BEGIN
            K := DISCRETE(TGTDEFAULTS(INDX).VECTOR,UTSEL);
            IF K LE 5 THEN
              BEGIN
                IF FACLIS(K).EMPTY THEN ALTGT := ALTGT -1
                ELSE
                  BEGIN
                    TGTSELECT :- RANDPICK(FACLIS(K));

```

```

        ALTGT := 0;
        END;
    END
    ELSE
        IF FACLISX(K).EMPTY THEN ALTGT := ALTGT -1
        ELSE
            BEGIN
                TGTSELECT :- RANDPICK(FACLISX(K));
                ALTGT := 0;
            END;
        END;
    END;
END
ELSE
    TGTSELECT :- IF K LE 5 THEN RANDPICK(FACLIS(K))
                ELSE RANDPICK(FACLISX(K));
END -- TGTSELECT --;

PROCEDURE GETS(F,CATC); REF(FACILITY) F;TEXT CATC;
BEGIN
    IF F IN BUILDING THEN
        BEGIN
            IF CATC.SUB(1,2) = "42" THEN
                F.INTO(THIS INSTALLATION QUA AIRBASE.FACLISX(8))
            ELSE IF CATC = "141D" THEN
                F.INTO(THIS INSTALLATION QUA AIRBASE.FACLISX(7))
            ELSE F.INTO(FBLDG);
        END
    ELSE IF F IN PETRO THEN F.INTO(FPOL)
    ELSE IF F IN UTILITY THEN
        BEGIN
            IF CATC.SUB(1,2) = "84" THEN
                F.INTO(THIS INSTALLATION QUA AIRBASE.FACLISX(9))
            ELSE F.INTO(FUTIL);
        END
    ELSE IF F IN TRANSPOR THEN F.INTO(FROAD)
    ELSE IF F IN SURFACE THEN
        BEGIN
            IF CATC.SUB(1,3) = "111" THEN F.INTO(FRUN)
            ELSE F.INTO(THIS INSTALLATION QUA AIRBASE.FACLISX(6));
        END;
    END -- GETS --;

FRAMES :- NEW HEAD;
BEGIN INTEGER I; FOR I:=6 STEP 1 UNTIL 9 DO FACLISX(I):-NEW HEAD; END;
INDX := 2;
END ** AIRBASE ** ;

```

Figure B-141

vv. Class AIRPOD.

i. Purpose. This is a subCLASS of AIRBASE that implements additional procedues associated with air ports of debarkation (APODs). When INSTALLATIONS are created by SEAC (see procedure BUILDMAP), airfields to be used as APODs and any associated service reception centers are identified.

ii. Class AIRPOD Skeleton.

Parameters and Attributes for Class AIRPOD

CLASS: AIRPOD	PREFIX: AIRBASE
PARAMETERS: none	
VARIABLE ATTRIBUTES:	
RECTRS GEOLOC[]	indicates those CENTERS that are used for reception centers
PROCEDURE ATTRIBUTES:	
GEOCTRS	checks list of candidate centers for valid service CENTER entry

Figure B-142

iii. Listing.

Listing of Class AIRPOD

```
AIRBASE CLASS AIRPOD;
BEGIN
  REF(GEOLOC) ARRAY RECTRS(1:4);
  PROCEDURE GEOCTRS(T);TEXT T;
    BEGIN INTEGER I,S;
      I :=1;
      WHILE I LE T.LENGTH DO
        IF T.SUB(I,1) = " " THEN
          I := I + 1
        ELSE
          BEGIN
            S := SVCSCK(T.SUB(I,1));
            IF S = 0 THEN
              BEGIN SETPOS(10);
                OUTTEXT("  APOD  SVC ERROR IN ");
                OUTTEXT(T); OUTIMAGE;
                I := I + 6;
              END
            ELSE
              BEGIN
```

```

RECTRS(S) :- GEOTAB.FIND(T.SUB(I+2,4));
IF RECTRS(S) == NONE THEN
  BEGIN
    OUTTEXT("--APOD--RECEP CTR UNDEFINED ");
    OUTTEXT(T.SUB(I,6));OUTIMAGE;
  END;
  I := I + 6;
END;
END;
END -- GEOCTRS --;
END ** AIRPOD **;

```

Figure B-143

ww. Class CAMP.

i. Purpose. This CLASS is used primarily for Army installations that do not fall within some of the other more specialized INSTALLTION CLASSES.

ii. Class CAMP Skeleton.

Parameters and Attributes for Class CAMP

CLASS: CAMP	PREFIX: INSTALLATION
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES: none	

Figure B-144

iii. Listing.

Listing of Class CAMP

```

INSTALLATION CLASS CAMP;
BEGIN
  INDX := 3;
END;

```

Figure B-145

xx. Class CENTER.

i. Purpose. This type of installation is a subCLASS of CAMP and is specifically intended to emulate installations that are used for unit and replacement personnel that are moving into and through the COMMZ. When a unit's personnel arrive in theater they generally move first to a reception center. They will then either move to a COMMZ location, or if in a combat unit to a staging base to marry up with their equipment before moving on to forward areas in the theater. To reflect the facility requirements at these installations, SEAC distinguishes between units that are merely passing through and those units that are likely to be permanently assigned. This is to more accurately reflect the reduced facility requirements to be attributed to the deploying units.

ii. Class CENTER Skeleton.

Parameters and Attributes for Class CENTER

CLASS: CENTER		PREFIX: CAMP
PARAMETERS: none		
VARIABLE ATTRIBUTES:		
TEMPOP	real	number of personnel in units that do not stay at the CENTER
MAXTEMPPOP	real	maximum temporary population up to that point in time
HOLDING	HEAD	set of units temporarily at the CENTER
RSDUNITS	HEAD	set of temporary units that arrived at the CENTER that day
PROCEDURE ATTRIBUTES:		
CHECKIN		processes incoming units (both permanent and temporary)
STATUS		prints permanent and temporary troop population figures

Figure B-146

iii. Listing.

Listing of Class CENTER

```
CAMP CLASS CENTER;
  BEGIN
  REAL TEMPOP,MAXTEMPPOP;
  REF(HEAD) HOLDING,RSDUNITS;

  PROCEDURE CHECKIN;
    BEGIN
    REF(AUNIT)A,T; REAL PERMS;
    FOR A :- RSDUNITS.FIRST WHILE A /= NONE DO
      BEGIN TEMPOP := TEMPOP + A.MEN; A.INTO(HOLDING); END;
    A :- HOLDING.FIRST;
    WHILE A /= NONE DO
      IF A.TIMEOUT <= TIME THEN
        BEGIN
          T :- A.SUC;
          TEMPOP := TEMPOP - A.MEN;
          IF A.DEST == NONE THEN
            A.OUT
          ELSE
            IF A.FWD THEN
              BEGIN
                IF A.DEST IS CENTER THEN
                  BEGIN
                    A.TIMEOUT := TIME + STAGETIME;
                    A.INTO(A.DEST QUA CENTER.RSDUNITS);
                    A.DEST :- NONE;
                  END
                ELSE
                  A.INTO(A.DEST.ARRIVALS);
                END
              ELSE
                A.INTO(A.DEST.ARRIVALS);
            END
          A :- T;
        END
      ELSE
        A :- A.SUC;
      END
    A :- ARRIVALS.FIRST;
    WHILE A /= NONE DO
      BEGIN
        DETRQMT(A.UTC,1.0);
        PERMS := PERMS + A.MEN;
        A.OUT;
        A :- ARRIVALS.FIRST;
      END;
    DETRQMT("PEOPLE",PERMS);
    POPULATION := POPULATION + PERMS;
    IF TEMPOP > MAXTEMPPOP THEN
      BEGIN
        DETRQMT("RS&D",TEMPPOP-MAXTEMPPOP);
        MAXTEMPPOP := TEMPOP;
      END
    END
```

```

        END;
    END -- CHECKIN --;

PROCEDURE STATUS(ASTA,ESTA,TSTA);BOOLEAN ASTA,ESTA,TSTA;
    BEGIN REF(TASK) TSK;
    OUTTEXT(" ==[ ");OUTTEXT(INAME);
    OUTCHAR('/');OUTINT(COUN,1);OUTCHAR('/');OUTINT(SRVC,1);OUTCHAR('/');
    OUTTEXT(" ] STATUS REPORT.");SETPOS(50);OUTTEXT(" POPULATION:");
    OUTFIX(POPULATION,0,7); OUTTEXT(" MAX RS&D:"); OUTFIX(MAXTEMPPOP,0,7);
    OUTTEXT(", SORTIES:");OUTINT(SORTIESRECVD,5);
    OUTTEXT(", SPF ATTACKS:");OUTINT(RCATK,5); OUTIMAGE;
    IF ASTA THEN ASSETS.TRAVERSE;
    OUTTEXT(" ENGR UNITS");OUTINT(ENGRS.CARDINAL,6);
    IF ESTA AND POOL /= NONE THEN POOL.TALLY(ENGRS,TRUE);
    OUTTEXT(" TASKS");OUTINT(JOBS.CARDINAL,6);
    IF TSTA THEN
        BEGIN
            TSK :- JOBS.FIRST;
            IF TSK == NONE THEN OUTIMAGE
                ELSE
                    WHILE TSK /= NONE DO BEGIN TSK.DUMP; TSK :- TSK.SUC; END;
            END
        ELSE
            OUTIMAGE;
    END -- STATUS --;

    HOLDING :- NEW HEAD;RSDUNITS :- NEW HEAD;
    END ** CENTER **;

```

Figure B-147

yy. Class PORT.

i. Purpose. As the name suggests this INSTALLATION subCLASS is used to define ports. (During the first application of SEAC, ports were not modeled to the same operational detail as were for example AIRBASEs. Future applications would no doubt place more emphasis on ports and demand that more detail be included regarding their special facility needs.)

ii. Class PORT Skeleton.

Parameters and Attributes for Class PORT

CLASS: PORT	PREFIX: INSTALLATION
PARAMETERS: none	
VARIABLE ATTRIBUTES: none	
PROCEDURE ATTRIBUTES: none	

Figure B-148

iii. Listing.

Listing of Class PORT

```
INSTALLATION CLASS PORT;  
  INDX := 4;
```

Figure B-149

zz. Class SEAPOD.

i. Purpose. Just as AIRBASE had a subCLASS to represent APODs, PORT has a subCLASS, SEAPOD, that can be used to designate reception centers for units that are arriving at the PORT, but are moving elsewhere in the theater.

ii. Class SEAPOD Skeleton.

Parameters and Attributes for Class SEAPOD

CLASS: SEAPOD	PREFIX: PORT
PARAMETERS: none	
VARIABLE ATTRIBUTES: RECTRS GEOLOC[]	
PROCEDURE ATTRIBUTES: GEOCTRS	

Figure B-150

iii. Listing.

Listing of Class SEAPOD

```
PORT CLASS SEAPOD;
BEGIN
  REF(GEOLOC) ARRAY RECTRS(1:4);
  PROCEDURE GEOCTRS(T);TEXT T;
    BEGIN INTEGER I,S;
    I :=-1;
    WHILE I LE T.LENGTH DO
      IF T.SUB(I,1) = " " THEN
        I := I + 1
      ELSE
        BEGIN
          S := SVCSCK(T.SUB(I,1));
          IF S = 0 THEN
            BEGIN SETPOS(10);
            OUTTEXT("__SPOD__SVC ERROR IN ");
            OUTTEXT(T); OUTIMAGE;
            I := I + 6;
            END
          ELSE
            BEGIN
              RECTRS(S) :- GEOTAB.FIND(T.SUB(I+2,4));
              IF RECTRS(S) == NONE THEN
                BEGIN
                  OUTTEXT("--SPOD--RECEP CTR UNDEFINED ");
                  OUTTEXT(T.SUB(I,6));OUTIMAGE;
                  END;
                I := I + 6;
              END;
            END;
          END;
        END -- GEOCTRS --;
      END ** SEAPOD **;
```

Figure B-151

aaa. Class ADA.

i. Purpose. This INSTALLATION subCLASS is intended to model air defense artillery sites. In the first application of SEAC, ADA sites were presumed to be permanent hardened locations that did not move. This was an artifact of the scenario used during SEAC's development. To the extent that future guidance foresees a different posture for ADA sites, this CLASS would have to be changed.

ii. Listing.

Listing of Class ADA

INSTALLATION CLASS ADA;
INDX := 5;

Figure B-152

bbb. Class HOSPITAL.

i. Purpose. (Hospital facilities are currently treated as belonging to the major INSTALLTION as opposed to being defined as a separate INSTALLATION.)

ii. Listing.

Listing of Class HOSPITAL

INSTALLATION CLASS HOSPITAL;
INDX := 6;

Figure B-153

ccc. Class DEPOT.

i. Purpose. DEPOTs currently contain no specialized procedural or data structure attributes beyond those derived from the prefix CLASS.

ii. Listing.

Listing of Class DEPOT

INSTALLATION CLASS DEPOT;
INDX := 7;

Figure B-154

ddd. Class ELEC.

i. Purpose. This CLASS is intended to represent the various electronic sites (radar, communications, etc.) that would be in the COMMZ. ELEC currently contains no specialized procedural or data structure attributes beyond those derived from the prefix CLASS. The factor file, however, does contain an entry for ELEC to assure that there are some baseline facilities (because assets seem to be frequently missing for smaller installations).

ii. Listing.

Listing of Class ELEC

INSTALLATION CLASS ELEC;
INDX := 8;

Figure B-155

7. Processes. The previous section of this annex described the many CLASSES used in SEAC which are passive, i.e. they have no internal ability to initiate actions. The PROCESSES described below (along with the main program) provide direction and initiate actions among themselves and all other objects in the runtime environment. Several of the CLASSES below refer the reader to more elaborate descriptions that appear in the main paper--further testament to their controlling function within SEAC.

a. Class RSD.

i. Purpose. This PROCESS controls the disposition of US units in the theater. It distinguishes between units entering the theater and those forward deployed; it identifies (to the degree that it can) those units that deploy to the COMMZ, those that deploy to the FCZ, and those that go to reception centers and then to COMMZ staging bases but are probably moving to the FCZ. (See the main paper section on Reception, Staging & Deployment.) An accurate representation of RS&D was a design goal of SEAC because of the inability of other automated planning systems to adequately treat unit movement through the theater.

ii. CLASS RSD Skeleton.

Parameters and Attributes for Class RSD

CLASS: RSD		PREFIX: PROCESS
PARAMETERS:		
ARVFILE	text	name of the file containing the Troop file
ECHO	boolean	if true then input record is written into the log file
VARIABLE ATTRIBUTES:		
TRP	INFILE	TROOP input file object
PS	INSTALLATION	POD INSTALLATIONs for units with POD -/- DEST
DS	INSTALLATION	destination INSTALLATION for the unit
P	INSTALLATION	used to find POD INSTALLATIONs
R	INSTALLATION	not used
S	INSTALLATION	used for destination INSTALLATIONs
PG	GEOLOC	the GEOLOC object associated with POD
C	CENTER	reception center defined for the unit's POD and service
CFAULT	CENTER	default (army) reception center
ASVC	character	character variable from SVC
TP	real	TIMEPOD's real variable equivalent
USTR	real	STR's real variable equivalent
ISVC	integer	SVC's integer variable equivalent
UTC	text	unit type code field on TROOP
FRQN	text	force requirements number field
POD	text	port of debarkation geoloc field
TIMEPOD	text	time arriving at POD field
DEST	text	destination geoloc field
STR	text	strength field
SVC	text	service indicator field
PROCEDURE ATTRIBUTES:		
ERROR		routine that formats RSD errors

Figure B-156

iii. Listing.

Listing of Class RSD.

```
PROCESS CLASS RSD(ARVFILE,ECHO);                                | ... RSD ...;
  TEXT ARVFILE;BOOLEAN ECHO;
  BEGIN
    REF(INFILE) TRP; REF(INSTALLATION) PS,DS,P,R,S; REF(GEOLOC) PG;
    REF(CENTER) C,CFAULT; CHARACTER ASVC; REAL TP,USTR; INTEGER ISVC;
    TEXT UTC,FRQN,POD,TIMEPOD,DEST,STR,SVC;

    PROCEDURE ERROR(MSG); TEXT MSG;
      BEGIN SETPOS(10);
        OUTTEXT("..RSD.. ");OUTTEXT(MSG);
        OUTTEXT(" FOR ENTRY -- ");OUTTEXT(TRP.IMAGE.SUB(1,47));
        OUTIMAGE;
        END;

    TRP :- NEW INFILE(ARVFILE);
    TRP.OPEN(BLANKS(80));
    TRP.INIMAGE;
    UTC :- TRP.IMAGE.SUB(19,5);          FRQN :- TRP.IMAGE.SUB(7,5);
    POD :- TRP.IMAGE.SUB(37,4);          TIMEPOD :- TRP.IMAGE.SUB(41,3);
    DEST :- TRP.IMAGE.SUB(24,4);         STR :- TRP.IMAGE.SUB(31,6);
    SVC :- TRP.IMAGE.SUB(47,1);

    WHILE NOT TRP.ENDFILE DO
      BEGIN
        TP :- TIMEPOD.SUB(1,3).GETREAL;
        ISVC :- SVCSCK(SVC.SUB(1,1));
        ASVC :- SVC.SUB(1,1).GETCHAR;
        USTR :- STR.SUB(1,6).GETREAL;
        IF TIME <= TP THEN PASSIVATE;
        IF ECHO THEN BEGIN OUTTEXT(">TRP:");OUTTEXT(TRP.IMAGE);OUTIMAGE;END;
        IF TP < 0.0 THEN
          BEGIN
            DS :- MATCH(GEOTAB.FIND(DEST),XUS,ISVC);
            IF DS == NONE THEN
              ERROR("IN PLACE UNIT DEST MISSING")
            ELSE
              NEW AUNIT(UTC,ASVC,FALSE,USTR,TIME,NONE).INTO(DS.ARRIVALS);
            END
          ELSE
            IF DEST = POD THEN
              BEGIN
                P:-MATCH(GEOTAB.FIND(DEST),XUS,ISVC);
                IF P == NONE THEN
                  ERROR("POD(=DEST) NOT FOUND")
                ELSE
                  NEW AUNIT(UTC,ASVC,FALSE,USTR,TIME,NONE).INTO(P.ARRIVALS);
                END
              ELSE
                BEGIN
```



```

PS :- C :- CFAULT :- NONE;
PG :- GEOTAB.FIND(POD);
IF PG == NONE THEN
    ERROR("POD GEOLOC NOT FOUND")
ELSE
    BEGIN
    P :- PG.SITES.FIRST;
    WHILE P /= NONE DO
        BEGIN
        IF P IS AIRPOD OR P IS SEAPOD THEN PS :- P;
        P :- P.SUC;
        END;
    INSPECT PS
    WHEN AIRPOD DO
        BEGIN
        C :- MATCH(RECTRS(ISVC),XUS,ISVC);
        IF C == NONE AND ISVC < XARMY THEN
            CFAULT:-MATCH(RECTRS(XARMY),XUS,XARMY);
        END
    WHEN SEAPOD DO
        BEGIN
        C :- MATCH(RECTRS(ISVC),XUS,ISVC);
        IF C == NONE AND ISVC < XARMY THEN
            CFAULT:-MATCH(RECTRS(XARMY),XUS,XARMY);
        END
    OTHERWISE ERROR("POD NOT DEFINED AS APOD/SPOD");
    IF C == NONE THEN
        BEGIN
        ERROR("NO SERVICE RECEPTION CTR");
        IF CFAULT /= NONE THEN
            BEGIN
            ERROR("  ^^ ARMY CTR SUBSTITUTED");
            C :- CFAULT;
            END;
        END;
    IF C /= NONE THEN
        BEGIN
        S :- MATCH(GEOTAB.FIND(DEST),XUS,ISVC);
        IF S == NONE THEN
            ERROR("DEST GEOLOC NOT DEFINED(ASSUME FCZ)");
        IF DEPLOY(FRQN) THEN
            BEGIN
            NEW AUNIT(UTC,ASVC,TRUE,USTR,RECEPTEIME+TIME,S)
                .INTO(C.RSDUNITS);
            IF NOT S IS CENTER THEN
                ERROR("DEPLOYING DEST IS NOT STAGING CTR");
            END
            ELSE
                NEW AUNIT(UTC,ASVC,FALSE,USTR,
                    RECEPTEIME+TIME,S).INTO(C.RSDUNITS);
            END;
        END;
    END;
    TRP.INIMAGE;

```

```

    END;

TRP.CLOSE;

END ** RSD **;

```

Figure B-157

b. Class CHANGE.

i. Purpose. This PROCESS permits the user to do three things: to account for possible COMMZ expansion or contraction; to indicate known facility requirements (projects) that the usual factors would not generate; and most importantly to assign, move, and change engineer units in the COMMZ. Changes to the areas considered within the COMMZ are handled by toggling MAPCELL variable AOR; if set false then the MAPCELL is not considered in normal requirements and capabilities calculations. Projects can be inserted into either INSTALLATIONS or MAPCELLS and would be things like pipelines, special port facilities, and other facilities planners identify. The manipulation of engineer units was another design goal within SEAC. Previous models of COMMZ engineering had been inflexible, units committed to an area remained there for the duration of the scenario. Many times there would be an abundance of engineers in one area (eg., base complex in CESP) and scarcity in an adjacent area with no mechanism to share. SEAC through its regionalization of the theater and the explicit treatment of engineer units (unit identities are preserved) through entries in the Order file allows better control of capability.

ii. Class CHANGE Skeleton.

Parameters and Attributes for Class CHANGE

CLASS: CHANGE		PREFIX: PROCESS
PARAMETERS:		
ORDFYL	text	name of file containing engineer unit & project directions
ECHO	boolean	if true then input record is written into the log file
VARIABLE ATTRIBUTES:		
ORDERS	INFILE	input file object for Orders
OTYM	text	effective time of order field
OTYPE	text	type of record: mapcell, project, or engineer unit order
OCELL	text	mapcell
CELLVAL	text	'T' if mapcell is in AOR, else 'F'
OUTC	text	engineer UTC field
ORDER	text	text buffer for ORDERS entries
ONAT	text	nationality field
OSVC	text	service field

OCAT	text	facility category code
OQUAN	text	quantity field
OUOM	text	unit of measure field
OTO	text	destination field (GEOLOC or MAPCELL index)
OFROM	text	location field (GEOLOC or MAPCELL INDEX)
OSTR	text	strength field
TYM	real	OTYM real variable equivalent
AMT	real	OQUAN real variable equivalent
UNITS	real	** not used **
MIX	integer	OCELL integer variable equivalent
RIX	integer	OTO/OFROM integer equivalent
NAT	integer	ONAT integer equivalent
SVC	integer	OSVC integer equivalent
RGN	REGION	** not used **
R	REGION	reference to object with RIX index
INX	INSTALLATION	reference for unit movements
INS	INSTALLATION	reference for project assignments
CFAC	FACILITY	** not used **
G	GEOLOC	reference to object that OTO indicates as a destination
EU	ENGRUNIT	references a new or existing object
EN	ENGRCAP	references entry in ENGRCAP tree for unit with OUTC id
CMP	COMPO	** not used **
P	PROPERTY	references object in installation's ASSET tree for required category code
CAT	CATCOD	identifies object in CATREE with OCAT category code

PROCEDURE ATTRIBUTES:		
ERROR		formats CHANGE error messages

Figure B-158

iii. Listing.

Listing of Class CHANGE

```
PROCESS CLASS CHANGE(ORDFYL,ECHO);
  TEXT ORDFYL;BOOLEAN ECHO;
  BEGIN
    REF(INFILE)ORDERS;
    TEXT OTYM,OTYPE,OCELL,CELLVAL,OUTC,ORDER;
    TEXT ONAT,OSVC,OCAT,OQUAN,OUOM,OTO,OFROM,OSTR;
    REAL TYM,AMT,UNITS; INTEGER MIX,RIX,NAT,SVC;
    REF(REGION)RGN,R; REF(INSTALLATION)INX,INS; REF(FACILITY)CFAC;
    REF(GEOLOC)G; REF(ENGRUNIT)EU; REF(ENGRCAP)EN;
    REF(COMPO)CMP; REF(PROPERTY)P; REF(CATCOD)CAT;

    PROCEDURE ERROR(T);TEXT T;
      BEGIN
        OUTTEXT("..CHANGE ERROR..");OUTTEXT(T);
        OUTTEXT("--");OUTTEXT(ORDER);OUTIMAGE;
        END -- ERROR --;

    ORDERS :- NEW INFILE(ORDFYL);
    ORDER :- BLANKS(80);
    ORDERS.OPEN(ORDER);
    ORDERS.INIMAGE;

    OTYM :- ORDER.SUB(1,5); OTYPE :- ORDER.SUB(10,1);
    ONAT :- ORDER.SUB(12,1); OSVC :- ORDER.SUB(14,1);

    OCELL :- ORDER.SUB(16,5); CELLVAL :- ORDER.SUB(26,1);
    OSTR:-ORDER.SUB(21,4);
    OUTC :- ORDER.SUB(16,5);OFROM :- ORDER.SUB(26,4);OTO :- ORDER.SUB(31,4);
    OCAT :- ORDER.SUB(16,4);OQUAN :- ORDER.SUB(22,8);OUOM :- ORDER.SUB(36,2);

    WHILE NOT ORDERS.ENDFILE DO
      BEGIN
        TYM := OTYM.SUB(1,5).GETREAL;
        NAT := COUNCK(ONAT.SUB(1,1));
        SVC := SVCSCK(OSVC.SUB(1,1));
        IF TYM <= TIME THEN
          BEGIN
            IF ECHO THEN BEGIN OUTTEXT(">EXO:");OUTTEXT(ORDER);OUTIMAGE;END;
            IF OTYPE = "U" THEN
              BEGIN
                EU :- NONE;
                IF OFROM = " " THEN
                  BEGIN
                    EN :- ECAPTAB.FIND(OUTC);
                    IF EN /= NONE THEN
                      EU :- NEW ENGRUNIT(EN,OSTR.SUB(1,4).GETREAL,TYM)
                    ELSE
                      ERROR("UNIDENTIFIED ENGR UTC ");
                  END
                END
              END
            END
          END
        END
      END
```

```

ELSE
IF LETTER(OFROM.SUB(1,4).GETCHAR) THEN
    BEGIN
    INX:-MATCH(GEOTAB.FIND(OFROM),NAT,SVC);
    IF INX == NONE THEN ERROR("FROM INSTAL NOT FND/MTCHD")
        ELSE EU :- GET(OUTC,INX.ENGRS);
    END
ELSE
    BEGIN
    EN:-ECAPTAB.FIND(OUTC);
    IF EN -/= NONE THEN
        BEGIN
        RIX:-OFROM.SUB(1,4).GETINT;
        R :- RGNSTRUC(EN.NATION).LOCGRN(RIX);
        IF R == NONE THEN
            ERROR("FROM REGION NOT FOUND")
        ELSE
            EU:-GET(OUTC,R.ENGRASSETS);
        END;
    END;
IF EU == NONE THEN
    ERROR("UNIT NOT IN FROM-LOC")
ELSE
    IF OTO = " " THEN
        EU.OUT
    ELSE
        IF LETTER(OTO.SUB(1,4).GETCHAR) THEN
            BEGIN
            G :- GEOTAB.FIND(OTO);
            INX:-MATCH(G,NAT,SVC);
            IF INX == NONE THEN
                BEGIN
                ERROR("TO INSTAL NOT FOUND/MATCHED");
                IF EU.UTCMSTR.SERVICE = XARMY THEN
                    BEGIN
                    IF G -/= NONE THEN
                        BEGIN
                        ERROR("^...PUT ARMY ENGR INTO
                            REGION");
                        EU.INTO(G.CELL.RGNLINK(NAT)
                            .ENGRASSETS);
                        END
                    ELSE
                        ERROR("-...GEOLOC NOT DEFINED");
                    END;
                END
            ELSE
                EU.INTO(INX.ENGRS);
            END
        ELSE
            BEGIN
            RIX:=OTO.SUB(1,4).GETINT;
            R :- RGNSTRUC(EU.UTCMSTR.NATION).LOCGRN(RIX);
            IF R == NONE THEN

```

```

                                ERROR("TO REGION NOT FOUND")
                                ELSE
                                EU.INTO(R.ENGRASSETS);
                                END;
END
ELSE
IF OTYPE = "M" THEN
    BEGIN
    MIX:=OCELL.SUB(1,5).GETINT;
    AREAS(MIX).AOR := IF CELLVAL = "T" THEN TRUE ELSE FALSE;
    END
    ELSE
    IF OTYPE = "P" THEN
        BEGIN
        AMT:= OQUAN.SUB(1,8).GETREAL;
        IF LETTER(OTO.SUB(1,4).GETCHAR) THEN
            BEGIN
            G :- GEOTAB.FIND(OTO);
            INS :- MATCH(G,NAT,SVC);
            IF INS == NONE THEN ERROR("EX-TASK GEOLOC/SERVICE")
            ELSE
                BEGIN
                P :- INS.ASSETS.FIND(OCAT);
                IF P == NONE THEN
                    BEGIN
                    P :- NEW PROPERTY(OCAT);
                    IF P.XREF(INS) THEN P.RANK(INS.ASSETS);
                    END;
                ELSE ERROR("COMPONENT NOT FOUND");
            END;
        END
        ELSE
        BEGIN
        MIX := OTO.SUB(1,4).GETINT;
        IF AREAS(MIX) /= NONE THEN
            BEGIN
            IF NOT AREAS(MIX).HASA(OCAT,NAT,SVC,
                                QUOM,AMT.XISTD)
                THEN
                ERROR(" MSR ASSET REJECTED");
            END
            ELSE
            ERROR(" MAPCELL INDEX");
            END;
        END
        ELSE ERROR(" BAD ORDER FORMAT ");
    ORDERS.INIMAGE;
    END
    ELSE
    PASSIVATE;
    END;
ORDERS.CLOSE;
OUTTEXT("#### ORDERS FILE HAS BEEN CLOSED");

```

OUTIMAGE;
END *** CHANGES ***;

Figure B-159

c. Class MISSION.

i. Purpose. A MISSION object is generated for each air attack generated by MISGEN. It indicates which planes are making the attack (GROUP), where they came from (ORIGIN) and where they also will return to, and what is being attacked (TGT). Along the way planes may be damaged or destroyed, and may or may not actually attack the target. The original concept for MISSION also included a feature to measure flight time to and from the target; this was to better measure turnaround time and the ability of the enemy to meet the sortie rates projected by threat assessments. The timing feature was not implemented in the first version of SEAC because of circumstances in the controlling scenario. (See the description of the AIR WAR in the main paper.)

ii. Class MISSION Skeleton.

Parameters and Attributes for Class MISSION.

CLASS: MISSION		PREFIX: PROCESS
PARAMETERS:		
MISCODE	integer	sequential number of the mission
TGT	INSTALLATION	target
TGTNAME	text	name of the target
ORIGIN	NKAIRBASE	home location of the group
SUMX	integer[]	vector used to record cumulative sortie information
VARIABLE ATTRIBUTES:		
GROUP	HEAD	set containing planes on mission
KILLED	HEAD	planes that are killed are transferred from GROUP to this set
DAMAGED	HEAD	damaged planes are put in this set
P	PLANE	reference to GROUP member
TEMP	PLANE	used to track objects in GROUP
TOTBOMBS	integer	** not used **
AIMAT	integer	** not used **
I	integer	indicates type index of PLANE
FLTIME	real	assumed to be 0 (i.e., not used)
PROCEDURE ATTRIBUTES:		
ASSIGN		initializes MISSION sets

Figure B-160

iii. Listing.

Listing of Class MISSION.

```

PROCESS CLASS MISSION(MISCODE,TGT,TGTNAME,ORIGIN,SUMX);           | MISSION;
REF(INSTALLATION)TGT;TEXT TGTNAME;INTEGER ARRAY SUMX;
REF(NKAIRBASE)ORIGIN;
INTEGER MISCODE;
BEGIN
REF(HEAD)GROUP,KILLED,DAMAGED;REF(PLANE)P,TEMP;
INTEGER TOTBOMBS,AIMAT,I;
REAL FLTIME;

PROCED ASSIGN;
  BEGIN GROUP :- NEW HEAD; KILLED :- NEW HEAD; DAMAGED :- NEW HEAD;
  END -- ASSIGN --;

HOLD(FLTIME);

```



```

P :- GROUP.FIRST;
WHILE P /= NONE DO
  BEGIN
    I := P.TYPE.PID;
    IF DRAW(HIT,UHIT) THEN
      BEGIN
        IF DRAW(PBEFOR,UBEFOR) THEN
          BEGIN
            TEMP :- P;
            P :- P.SUC;
            IF DRAW(PKILL,UKILL) THEN
              BEGIN TEMP.INTO(KILLED); SUMX(I,3):=SUMX(I,3)+1; END
            ELSE
              BEGIN TEMP.INTO(DAMAGED);SUMX(I,5):=SUMX(I,5)+1;END;
            END
          ELSE
            BEGIN
              IF DRAW(PSUC,USUC) THEN
                BEGIN P.ATTACK(TGT);SUMX(I,7):=SUMX(I,7)+1;END;
              TEMP :- P;
              P :- P.SUC;
              IF DRAW(PKILL,UKILL) THEN
                BEGIN TEMP.INTO(KILLED);SUMX(I,4):=SUMX(I,4)+1;END
              ELSE
                BEGIN TEMP.INTO(DAMAGED);SUMX(I,6):=SUMX(I,6)+1;END;
              END;
            END
          END
        ELSE
          BEGIN
            IF DRAW(PSUC,USUC) THEN
              BEGIN P.ATTACK(TGT);SUMX(I,7):=SUMX(I,7)+1;END;
            P :- P.SUC;
            END;
          END;
        END;
      HOLD(FLTIME);
      OUTTEXT( "MISSION REPORT  ");OUTINT(MISCODE,4);SETPOS(POS+3);
      OUTTEXT(ORIGIN.ID);
      OUTTEXT(" TO ");OUTTEXT(TGTNAME);
      OUTINT(GROUP.CARDINAL,10);
      OUTINT(KILLED.CARDINAL,10);
      OUTINT(DAMAGED.CARDINAL,10);OUTIMAGE;
      P :- GROUP.FIRST;
      WHILE P /= NONE DO
        BEGIN
          TEMP :- P.SUC;
          IF P.TYPE IS BMBR THEN
            P.INTO(ORIGIN.BOMBERS)
          ELSE
            P.INTO(ORIGIN.FIGHTERS);
          P :- TEMP;
          END;
        P :- DAMAGED.FIRST;
        WHILE P /= NONE DO
          BEGIN

```

```

P.REPAIRTIME := TIME + NEGEXP(XREP,UREP);
P.INTO(ORIGIN.REPAIR);
P :- DAMAGED.FIRST;
END;
END ** MISSION **;

```

Figure B-161

d. Class MISGEN.

i. Purpose. This CLASS generates air attacks (instances of CLASS MISSION) based on instructions read from an external air threat file. The file contains scenario data collected from applicable threat assessments which provides a likely threat attack strategy. Entries in the file give: time, planes, home field, target location, and, optionally, facility target priorities. It must be emphasized that threat attacks follow the preplanned directions found in the file. SEAC does not model enemy targeting, thus the success or failure of prior attacks does not influence how current attacks are developed. The file directives control.

ii. Class MISGEN Skeleton.

Parameters and Attributes for Class MISGEN.

CLASS: MISGEN		PREFIX: PROCESS
PARAMETERS:		
TMSNS	text	name of file containing the air threat
ECHO	boolean	if true then input record is written into the log file
VARIABLE ATTRIBUTES:		
ORD	text	input buffer for air threat file
DAY	text	field for day of attack
STRIKE	text	field for hour of attack
TGTLOC	text	target label (geoloc) field
ORIGIN	text	home base field (geoloc or other)
BMRS	text	bombers in mission field
FBRS	text	fighter bombers in mission field
RDY	text	operational readiness field
TGT	INSTALLATION	target selected
ORI	NKAIRBASE	threat airbase source for mission
M	MISSION	references object
WAVE	HEAD	MISSIONs are collected in this set
TGTGEO	GEOLOC	reference to target's GEOLOC
ORDERS	INFILE	input file object for air threat
MSNTGTS	TGTPREF	reference to facility targeting directive
AIRSUM	integer[]	array to store summary statistics
NOB	integer	BMRS integer variable equivalent
NOF	integer	FBRS integer variable equivalent
BAVAIL	integer	count of bombers available for the MISSION
FAVAIL	integer	count of fighter/bombers available for the MISSION
I	integer	loop counter
J	integer	loop counter
WSTR	integer	working time (strike)
CSTR	integer	current hour (strike)
MISCNT	integer	used to count MISSIONs flown
UALO	integer	random number seed for tgt aloc
WDAY	real	day into scenario for attack
RDYRATE	real	operational readiness rate
PROCEDURE ATTRIBUTES:		
RELEASE		clears the wave set
ALLOCATE		transfers PLANES from base to MISSION.GROUP and assigns an aimpoint to each

Figure B-162

iii. Listing.

Listing of Class MISGEN.

```
PROCESS CLASS MISGEN(TMSNS,ECHO); TEXT TMSNS; BOOLEAN ECHO;      | MISGEN;
BEGIN
TEXT ORD, DAY, STRIKE, TGTLOC, ORIGIN, BMRS, FBRS, RDY;
REF(INSTALLATION) TGT;      REF(NKAIRBASE) ORI;      REF(MISSION)M;
REF(HEAD)WAVE;      REF(GEOLOC)TGTGEO;      REF(INFILE) ORDERS;
REF(TGTPREF) MSNTGTS;      INTEGER ARRAY AIRSUM(1:2,1:7);
INTEGER NOB, NOF, BAVAIL, FAVAIL, I, J, WSTR, CSTR, MISCNT, UALO;
REAL WDAY, RDYRATE;

PROCEDURE RELEASE;
BEGIN
REF (MISSION)X;
X :- WAVE.FIRST;
WHILE X /= NONE DO
    BEGIN ACTIVATE X; X.OUT; X :- WAVE.FIRST; END;
END;

PROCEDURE ALLOCATE(NPLNS, PLQ, MISQ, TGTMLPT);
INTEGER NPLNS; REF(HEAD) PLQ, MISQ; REF(TGTPREF) TGTMLPT;
INSPECT TGTMLPT DO
    WHILE NPLNS > 0 DO
        BEGIN
            PLQ.FIRST QUA PLANE.AIMPOINT:-DISCRETE(VECTOR,UALO);
            PLQ.FIRST.INTO(MISQ);
            NPLNS := NPLNS - 1;
        END;

UALO := 32419;
WAVE :- NEW HEAD;
ORD :- BLANKS(80);
ORDERS :- NEW INFILE(TMSNS);
ORDERS.OPEN(ORD);

DAY :- ORD.SUB(1,2); STRIKE :- ORD.SUB(4,2); RDY :- ORD.SUB(31,5);
TGTLOC :- ORD.SUB(7,4); ORIGIN :- ORD.SUB(12,4);
BMRS :- ORD.SUB(17,4); FBRS :- ORD.SUB(22,4);
ORDERS.INIMAGE;
WDAY := DAY.SUB(1,2).GETREAL;
RDYRATE := RDY.SUB(1,5).GETREAL;
IF ECHO THEN BEGIN OUTTEXT(">AIR:");OUTTEXT(ORD);OUTIMAGE; END;
WHILE NOT ORDERS.ENDFILE DO
    BEGIN
        WDAY := DAY.SUB(1,2).GETREAL;
        IF WDAY GT TIME THEN
            BEGIN
                OUTIMAGE;
                RDYRATE := RDY.SUB(1,5).GETREAL;
                CSTR:= -1 ;
            END;
        WSTR := STRIKE.SUB(1,2).GETINT;
```

```

IF WSTR NE CSTR THEN
  BEGIN
    RELEASE;
    CSTR:=WSTR;
    REACTIVATE CURRENT AT WDAY + CSTR/24.0;
    IF ECHO THEN BEGIN OUTTEXT(">AIR:");OUTTEXT(ORD);OUTIMAGE; END;
    OUTTEXT(" NEW STRIKE BEGUN ");OUTTEXT(STRIKE);OUTIMAGE;
    NKAB.AOBPOOL(RDYRATE);
    END;

TGTGEO :- GEOTAB.FIND(TGTLOC);
TGT :- IF TGTGEO /= NONE THEN TGTGEO.SITES.FIRST ELSE NONE;
IF TGT /= NONE THEN
  BEGIN
    ORI :- NKAB.FIND(ORIGIN);
    IF ORI /= NONE THEN
      BEGIN
        MISCNT := MISCNT + 1;
        M :- NEW MISSION(MISCNT,TGT,TGTGEO.GEOLAB,ORI,AIRSUM);
        M.ASSIGN;
        ORDERS.SETPOS(40);
        IF ORDERS.INCHAR = '*' THEN
          BEGIN
            MSNTGTS :-NEW TGTPREF(IF TGT IS AIRBASE THEN 9 ELSE 5);
            INSPECT MSNTGTS DO
              FOR J:=1 STEP 1 UNTIL NIN DO
                VECTOR(J):=ORDERS.INREAL;
          END
        ELSE
          MSNTGTS:-TGTDEFAULTS(TGT.INDX);
        IF BMRS NE BLANKS(4) THEN
          BEGIN
            NOB := BMRS.SUB(1,4).GETINT;
            AIRSUM(1,1) := AIRSUM(1,1) + NOB;
            BAVAIL := ORI.BOMBERS.CARDINAL;
            IF BAVAIL LT NOB THEN
              BEGIN
                OUTTEXT("BOMBER SHORTAGE FOR TARGET ");
                OUTTEXT(TGTLOC);OUTIMAGE;
                NOB := BAVAIL;
              END;
            ALLOCATE(NOB,ORI.BOMBERS,M.GROUP,MSNTGTS);
            AIRSUM(1,2) := AIRSUM(1,2) + NOB;
          END;
        IF FBRs NE BLANKS(4) THEN
          BEGIN
            NOF := FBRs.SUB(1,4).GETINT;
            AIRSUM(2,1) := AIRSUM(2,1) + NOF;
            FAVAIL := ORI.FIGHTERS.CARDINAL;
            IF FAVAIL LT NOF THEN
              BEGIN
                OUTTEXT("FIGHTER SHORTAGE FOR TARGET ");
                OUTTEXT(TGTLOC);OUTIMAGE;
                NOF := FAVAIL;
              END;
            END;

```

```

        END;
        ALLOCATE(NOF,ORI.FIGHTERS,M.GROUP,MSNTGTS);
        AIRSUM(2,2) := AIRSUM(2,2) + NOF;
        END;
        ACTIVATE M;
        M.INTO(WAVE);
        END
        ELSE
        BEGIN
        OUTTEXT("MISSING ORIGIN BASE ");
        OUTTEXT(ORIGIN);OUTIMAGE;
        END;
    END
    ELSE
    BEGIN
    OUTTEXT("MISSING TARGET ");OUTTEXT(TGTLOC);
    OUTIMAGE;
    END;
    ORDERS.INIMAGE;
    END;
    RELEASE;
    HOLD(1);
    NKAB :- NONE;      | CAUSES RELEASE OF AIR MODULE DATA STORAGE;
    OUTTEXT("+++ AIR WAR MISSIONS -"); OUTINT(MISCNT,5);OUTIMAGE;
    SETPOS(31);OUTTEXT("KILLED          DAMAGED");OUTIMAGE;
    OUTTEXT("      SCHED      ACTUAL      PRE      POST");
    OUTTEXT("      PRE      POST      ATK");
    OUTIMAGE;
    FOR I :- 1 STEP 1 UNTIL 2 DO
        BEGIN
        FOR J :- 1 STEP 1 UNTIL 7 DO
            OUTINT(AIRSUM(I,J),10);
        OUTIMAGE;
        END;
    ORDERS.CLOSE;
    END ** MISGEN ** ;

```

Figure B-163

e. Class REARAXN.

i. Purpose. This CLASS complements MISGEN by generating rear actions that inflict damage but are performed by enemy units that are on the ground rather than in the air. The purpose of REARAXN is to generate reasonable levels of facility damage within the capability and likely distribution of enemy commando or infiltration units. The success of enemy ground forces is largely dependent on various probabilities: that the infiltrating unit is not destroyed enroute, that it reaches its scheduled target, that it is successful in finding appropriate targets, and last that although it did not initially find its target it was not destroyed by COMMZ security forces. Friendly forces are not played other than in an implicit sense in the probability of an infiltration unit being killed. Since the attribute of the units of most concern is its ability to damage facilities, units are defined by the amount of

ordnance and explosives that it can carry and use to destroy facilities. It is important to note that once a unit attacks a target it is dropped from further consideration because it will have exhausted all the known ordnance that it carries in the attack. See the description of the Ranger/Commando logic in the main paper.

ii. Class REARAXN Skeleton.

Parameters and Attributes for Class REARAXN.

CLASS: REARAXN		PREFIX: PROCESS
PARAMETERS:		
RCFYL	text	name of file containing ground attack threat
ECHO	boolean	if true then input record is written into the log file
VARIABLE ATTRIBUTES:		
RCTHREAT	INFILE	input threat object for ground 1st part defines teams 2nd part lists attack scenario
S	SPFUNIT	reference variable
T	SPFUNIT	reference variable
INSTGT	INSTALLATION	target (installlation)
GTGT	GEOLOC	GEOLOC that contains target
ARTGT	MAPCELL	target (installations or MSRs)
SURVRC	HEAD	set containing surviving units
TEAMS	HEAD	list of defined unit types
TYPORD	ORDNANCE	points to entry in the ordnance tab
TM	TEAMORG	refers to unit being defined
INSRTMEANS	TEAMORG	refers to unit in attack directive
GROUP	text	input buffer for RCTHREAT
TIMEFLD	text	insertion time field
SIZE	text	size of attack (#units) field
GEOTGT	text	geoloc target (installation) field
AREATGT	text	area target (i.e. mapcell) field
TMNAME	text	unit's id label field (definition)
INSRTYPE	text	unit's id label field (scenario)
COLONE	text	a 'T' in this field indicates that it is to be interpreted as a unit definition record
ATTRATE	real	P(unit destroyed en route)
THRUTGT	real	P(gets to installation(commits))
PRCATTRIT	real	P(surviving unit is destroyed)
PRCATK	real	P(surviving unit attacks that day)
TYM	real	TIMEFLD real variable equivalent
URCATTRIT	integer	random seed
URCATK	integer	random seed
UATT	integer	random seed
UTHRU	integer	random seed
I	integer	loop counter

NCRAFT	integer	** not used **
THRTCNT	integer	number of type units in attack
RCSUM	integer[]	summary statistics for ground attacks

PROCEDURE ATTRIBUTES:		
REARBATTLE		the body of REARAXN deals with direct attacks by inserted units. This routine takes units in SURVRC to see if they are killed or if they attack a facility.

Figure B-164

iii. Listing.

Listing of Class REARAXN.

```

PROCESS CLASS REARAXN(RCFYL,ECHO); TEXT RCFYL;BOOLEAN ECHO;
  BEGIN
    REF(INFILE)RCTHREAT;      REF(SPFUNIT)S,T;
    REF(INSTALLATION)INSTGT;  REF(GEoloc)GTGT;      REF(MAPCELL)ARTGT;
    REF(HEAD) SURVRC,TEAMS;    REF(ORDNANCE)TYPORD;  REF(TeAMORG)TM,INSRTMEANS;
    TEXT GROUP,TIMEFLD,SIZE,GEOTGT,AREATGT,TMNAME,INSRTYPE,COLONE;
    REAL ATTRATE,THRUTGT,PRCATTRIT,PRCATK,TYM;
    INTEGER URCATTRIT,URCATK,UATT,UTHRU,I,NCRAFT,THRTCNT;
    INTEGER ARRAY RCSUM(1:8);

  PROCEDURE REARBATTLE;
    BEGIN
      S :- SURVRC.FIRST;
      WHILE S -/= NONE DO
        IF DRAW(PRCATTRIT,URCATTRIT) THEN
          BEGIN T:-S.SUC; S.OUT; S:-T; RCSUM(6):=RCSUM(6) + 1; END
        ELSE
          IF DRAW(PRCATK,URCATK) THEN
            BEGIN
              T:-S.SUC;
              IF ATKLOG THEN OUTTEXT("---SPF ATTACK: ");
              IF S.TORG.SABOTAGE(S.LOCALE) THEN
                BEGIN
                  S.OUT;
                  RCSUM(7) := RCSUM(7) + 1;
                END
              ELSE
                BEGIN
                  GTGT:-RANDPICK(S.LOCALE.GEosET);
                  IF GTGT == NONE THEN INSTGT :- NONE
                ELSE

```



```

        INSTGT :- RANDPICK(GTGT.SITES);
    IF INSTGT /= NONE THEN
        BEGIN
            S.TORG.ATK(INSTGT);
            S.OUT;
            RCSUM(8) := RCSUM(8) + 1;
            END
        ELSE
            S.OUT;
        END;
    IF ATKLOG THEN OUTIMAGE;
    S :- T;
    END;
END -- REARBATTLE --;

RCTHREAT :- NEW INFILE(RCFYL);
GROUP :- BLANKS(80);
RCTHREAT.OPEN(GROUP);

RCTHREAT.INIMAGE;
ATTRATE := RCTHREAT.INREAL;    UATT := RCTHREAT.ININT;
RCTHREAT.INIMAGE;
THRUTGT := RCTHREAT.INREAL;    UTHRU := RCTHREAT.ININT;
RCTHREAT.INIMAGE;
PRCATTRIT := RCTHREAT.INREAL;  URCATTRIT := RCTHREAT.ININT;
RCTHREAT.INIMAGE;
PRCATK := RCTHREAT.INREAL;     URCATK := RCTHREAT.ININT;
RCTHREAT.INIMAGE;
LBSTGT := RCTHREAT.INREAL;
RCTHREAT.INIMAGE;

TEAMS :- NEW HEAD;
COLONE:-GROUP.SUB(1,1);    TMNAME:-GROUP.SUB(6,5);
WHILE COLONE = "T" DO
    BEGIN
        IF ECHO THEN BEGIN OUTTEXT(">TEM:");OUTTEXT(GROUP);OUTIMAGE; END;
        TM :- NEW TEAMORG(TMNAME);
        TM.INTO(TEAMS);
        RCTHREAT.SETPOS(20);
        WHILE RCTHREAT.MORE DO
            BEGIN
                TYPORD:-ORDTAB.FIND(RCTHREAT.INTXT(5));
                IF TYPORD /= NONE THEN
                    NEW MUNITION(TYPORD,RCTHREAT.ININT).INTO(TM.ARMS)
                ELSE
                    BEGIN
                        IF NOT ECHO THEN OUTTEXT(RCTHREAT.IMAGE.SUB(1,RCTHREAT.POS))
                        ELSE SETPOS(RCTHREAT.POS);
                        OUTTEXT("^..SPF..MUNITION NOT FOUND--TM CENSORED");OUTIMAGE;
                        RCTHREAT.SETPOS( RCTHREAT.LENGTH + 1);
                    END;
            END;
        END;

        RCTHREAT.INIMAGE;
    
```

```

END;

SURVRC :- NEW HEAD;
TIMEFLD :- GROUP.SUB(1,5);      INSRTYPE :- GROUP.SUB(7,5);
SIZE :- GROUP.SUB(13,5);        GEOTGT :- GROUP.SUB(21,4);
AREATGT :- GROUP.SUB(27,4);
WHILE NOT RCTHREAT.ENDFILE DO
  BEGIN
    INSRTMEANS :- NONE; ARTGT :- NONE;
    TYM := TIMEFLD.SUB(1,5).GETREAL;
    IF TYM <= TIME THEN
      BEGIN
        IF ECHO THEN BEGIN OUTTEXT(">SPF:");OUTTEXT(GROUP);OUTIMAGE; END;
        IF GEOTGT = " " THEN
          BEGIN
            ARTGT :- AREAS(AREATGT.SUB(1,4).GETINT);
            GTGT :- NONE;
          END
        ELSE
          BEGIN
            GTGT :- GEOTAB.FIND(GEOTGT);
            IF GTGT /= NONE THEN ARTGT :- GTGT.CELL;
          END;
        IF ARTGT -/- NONE THEN
          BEGIN
            TM:- TEAMS.FIRST;
            WHILE TM -/- NONE DO
              IF TM.TID = INSRTYPE THEN
                BEGIN INSRTMEANS:-TM;TM:-NONE;END
              ELSE TM:-TM.SUC;
            IF INSRTMEANS == NONE THEN
              BEGIN
                OUTTEXT("..SPF.. DELIVERY MEANS ERROR ");
                OUTIMAGE;
              END
            ELSE
              BEGIN
                THRTCNT := SIZE.SUB(1,5).GETINT;
                RCSUM(1) := RCSUM(1) + THRTCNT;
                FOR I := 1 STEP 1 UNTIL THRTCNT DO
                  IF NOT DRAW(ATTRATE,UATT) THEN
                    BEGIN
                      IF GTGT == NONE THEN
                        BEGIN
                          NEW SPFUNIT(INSRTMEANS,ARTGT).INTO(SURVRC);
                          RCSUM(4) := RCSUM(4) + 1;
                        END
                      ELSE
                        IF DRAW(THRUTGT,UTHRU) THEN
                          BEGIN
                            INSTGT :- RANDPICK(GTGT.SITES);
                            IF INSTGT /= NONE THEN
                              BEGIN
                                IF ATKLOG THEN

```

```

        OUTTEXT("---SPF* ATTACK: ");
        INSRTMEANS.ATK(INSTGT);
        RCSUM(3) := RCSUM(3) + 1;
        IF ATKLOG THEN OUTIMAGE;
        END;
    END
    ELSE
    BEGIN
        NEW SPFUNIT(INSRTMEANS,ARTGT).INTO(SURVRC);
        RCSUM(5) := RCSUM(5) + 1;
        END;
    END
    ELSE
    RCSUM(2) := RCSUM(2) + 1;
    END;
END;
END
ELSE
BEGIN
    OUTTEXT("...SPF...TARGET NOT FOUND IN ");
    OUTTEXT(GROUP);OUTIMAGE;
    END;
RCTHREAT.INIMAGE;
END
ELSE
BEGIN
    REARBATTLE;
    HOLD(1);
    END;
END;

END;

RCTHREAT.CLOSE;
OUTTEXT("      VIII.A.  RANGER/COMMANDO FILE CLOSED AT TIME ");
OUTFIX(TIME,2,6);
OUTIMAGE;
WHILE NOT SURVRC.EMPTY DO
    BEGIN
        REARBATTLE;
        HOLD(1);
        END;
    OUTTEXT("      VIII.B.  RANGER COMMANDO THREAT EXHAUSTED AT TIME ");
    OUTFIX(TIME,2,6);OUTIMAGE;OUTIMAGE;
    OUTTEXT("RANGER COMMANDO SUMMARY:");
    OUTTEXT(" PLAND  X-IN  I1ATK  AASG  ADFLT  X-GND  MSR  I2ATK");
    OUTIMAGE;
    SETPOS(25); FOR I := 1 STEP 1 UNTIL 8 DO OUTTEXT(" -----"); OUTIMAGE;
    SETPOS(24); FOR I := 1 STEP 1 UNTIL 8 DO OUTINT(RCSUM(I),7); OUTIMAGE;
    OUTIMAGE;
    END *** REARAXN ***;

```

Figure B-165

8. Procedures -- Functions. The following procedures are those that are called more than once in a typical execution of SEAC. Most are what are usually referred to as functions, in that they usually return a value or object

reference. Exceptions are procedures DIARY and WORK which perform specific operations and are called once each simulated day.

a. Procedure RANDPICK.

i. Purpose. This procedure randomly selects an object from the set of objects passed to it as an argument. If the set, Q, is empty then the value returned is NONE.

ii. Procedure RANDPICK Skeleton.

Parameters and Local Variables for Procedure RANDPICK.

PROCEDURE: RANDPICK		TYPE: REF(LINK)
PARAMETERS:		
Q	HEAD	a set of objects
LOCAL VARIABLE:		
L	LINKAGE	pointer to set objects
I	integer	loop variable
M	integer	indicates the Mth item is picked
LOCAL PROCEDURE: none		

Figure B-166

iii. Listing.

Listing of Procedure RANDPICK.

```
REF(LINK) PROCEDURE RANDPICK(Q); REF(HEAD)Q;
BEGIN
REF(LINKAGE)L; INTEGER I,M;
IF NOT Q.EMPTY THEN
BEGIN
L :- Q; M :=RANDINT(1,Q.CARDINAL,PICKSEED);
FOR I := 1 STEP 1 UNTIL M DO L :- L.SUC;
RANDPICK :- L QUA LINK;
END;
END -- RANDPICK --;
```

Figure B-167

b. Procedure CATOFAC.

i. Purpose. This procedure takes a JCS category code (eg., 141B), and creates the FACILITY subCLASS object with which it is associated with internal to SEAC. The routine is setup to process a candidate similar to a binary search procedure. It is presently coded into a function but an alternative, and perhaps more efficient random access, or table lookup method may be preferable.

ii. Procedure CATOFAC Skeleton.

Parameters and Local Variables for Procedure CATOFAC.

PROCEDURE: CATOFAC		TYPE: REF(FACILITY)
PARAMETERS:		
PRP	PROPERTY	reference to installations real property table entry for catcode
CT	COMPO	reference to catcode entry
MAG	real	size of the facility
CTCODE	text	4 character JCS category code
CS	character	identifies construction standard
LOCAL VARIABLE:		
ONE	text	1st character of catcode
TWO	text	first 2 characters of catcode
THREE	text	first three characters of catcode
FOURTH	character	4th character of catcode
C	FACILITY	internal pointer to new FACILITY object
LOCAL PROCEDURE: none		

Figure B-168

iii. Listing.

Listing of Procedure CATOFAC.

```
REF(FACILITY) PROCEDURE CATOFAC(PRP,CT,MAG,CTCODE,CS);
  REF(COMPO)CT;REAL MAG;REF(PROPERTY)PRP;TEXT CTCODE;CHARACTER CS;
  BEGIN
    TEXT ONE,TWO,THREE;CHARACTER FOURTH;REF(FACILITY)C;

    ONE:-CTCODE.SUB(1,1); TWO:-CTCODE.SUB(1,2); THREE:-CTCODE.SUB(1,3);
    FOURTH := CTCODE.SUB(4,1).GETCHAR;
    IF ONE < "5" THEN
      BEGIN
        IF ONE < "2" THEN
          BEGIN
            IF TWO < "13" THEN
              BEGIN
                IF THREE = "111" THEN C:-NEW RUNWAY(MAG,PRP,CT,CS)
                ELSE
                  IF THREE < "120" THEN C:-NEW PAVEMENT(MAG,PRP,CT,CS)
                  ELSE
                    IF THREE<"124" THEN C:-NEW POLFACIL(MAG,PRP,CT,CS)
                    ELSE
                      IF THREE < "125" THEN C:-NEW TANK(MAG,PRP,CT,CS)
                      ELSE
                        IF FOURTH = 'A' THEN C:-NEW
                        PIPE(MAG,PRP,CT,CS)
                        ELSE C:-NEW POLFACIL(MAG,PRP,CT,CS);
              END
            ELSE
              IF TWO < "14" THEN
                BEGIN
                  IF THREE = "136" THEN C:-NEW POWER(MAG,PRP,CT,CS)
                  ELSE C:-NEW COMO(MAG,PRP,CT,CS);
                END
              ELSE
                IF TWO < "15" THEN
                  BEGIN
                    IF THREE = "141" THEN C:-NEW OPNS(MAG,PRP,CT,CS)
                    ELSE C:-NEW REVETMENT(MAG,PRP,CT,CS);
                  END
                ELSE
                  IF THREE="153" THEN C:-NEW STORAGE(MAG,PRP,CT,CS)
                  ELSE
                    IF THREE="156" THEN C:-NEW OPNS(MAG,PRP,CT,CS)
                    ELSE C:-NEW PIER(MAG,PRP,CT,CS);
                  END
                END
              ELSE
                IF ONE < "3" THEN C:-NEW SHOP(MAG,PRP,CT,CS)
                ELSE
                  IF TWO LT "42" THEN C:-NEW TANK(MAG,PRP,CT,CS)
                  ELSE C:-NEW STORAGE(MAG,PRP,CT,CS);
                END
              ELSE
                IF ONE = "5" THEN C:-NEW MEDICAL(MAG,PRP,CT,CS)
                ELSE
```

```

IF ONE < "8" THEN
  BEGIN
    IF ONE = "6" THEN C:-NEW ADMIN(MAG,PRP,CT,CS)
    ELSE C:-NEW QTRS(MAG,PRP,CT,CS);
  END
ELSE
  IF TWO < "84" THEN
    BEGIN
      IF TWO = "81" THEN C:-NEW POWER(MAG,PRP,CT,CS)
      ELSE C:-NEW WASTE(MAG,PRP,CT,CS);
    END
  ELSE
    IF TWO < "85" THEN C:-NEW WATER(MAG,PRP,CT,CS)
    ELSE
      IF TWO<"86" THEN
        BEGIN
          IF THREE = "851" THEN
            C:-NEW ROAD(MAG,PRP,CT,CS,FOURTH)
          ELSE
            C:-NEW HARDSTAND(MAG,PRP,CT,CS);
        END
      ELSE
        IF THREE = "860" THEN
          C:-NEW RR(MAG,PRP,CT,CS,FOURTH)
        ELSE
          IF THREE = "872" THEN
            C:-NEW REVETMENT(MAG,PRP,CT,CS);
          IF C /= NONE THEN CATOFAC :- C
          ELSE
            BEGIN
              OUTTEXT("..CATOFAC..FALL THROUGH ");
              OUTTEXT(CTCODE);OUTIMAGE;
            END;
          END -- CATOFAC --;

```

Figure B-169

c. Procedure PRIORANK.

i. Purpose. This function converts the letter character used to designate task priorities (eg., 'V' stands for vital) into its relative numerical ranking ('V' -> 1, 'C' -> 2, etc.).

ii. Procedure PRIORANK Skeleton.

Parameters and Local Variables for Procedure PRIORANK.

PROCEDURE: PRIORANK	TYPE: integer
PARAMETERS:	
T	character priority character 'V', 'C', etc.
LOCAL VARIABLE: none	
LOCAL PROCEDURE: none	

Figure B-170

iii. Listing.

Listing of Procedure PRIORANK.

```
INTEGER PROCEDURE PRIORANK(T); CHARACTER T;  
PRIORANK := IF T = 'V' THEN 1  
            ELSE IF T = 'C' THEN 2  
            ELSE IF T = 'E' THEN 3  
            ELSE IF T = 'N' THEN 4  
            ELSE 99;
```

Figure B-171

d. Procedure RATE

i. Purpose. This routine returns (if it exists) a reference to a POLICY object for a given country, type of work (presently only construction or repair), and facility (for repair) or installation (for construction) internal index.

ii. Procedure RATE Skeleton.

Parameters and Local Variables for Procedure RATE.

PROCEDURE: RATE		TYPE: REF(POLICY)
PARAMETERS:		
COUN	integer	country index
TGRP	character	task group ('C' -> construction)
INST	integer	installation index
FAC	integer	facility index
LOCAL VARIABLE: none		
LOCAL PROCEDURE: none		

Figure B-172

iii. Listing.

Listing of Procedure RATE.

```

REF(POLICY) PROCEDURE RATE(COUN,TGRP,INST,FAC);
CHARACTER TGRP; INTEGER INST,COUN,FAC;
BEGIN
IF TGRP = 'C' THEN
  BEGIN
  IF INST <> 0 THEN RATE :- CNSTRPLCY(INST,COUN);
  END
  ELSE
  IF TGRP = 'D' THEN
    RATE :- DAMPLCY(FAC,COUN)
  ELSE
    IF TGRP = 'R' THEN
      RATE :- RSTRPLCY(INST,COUN)
    ELSE
      IF TGRP = 'M' THEN
        RATE :- MNTNPLCY(FAC,COUN)
      ELSE
        BEGIN
        OUTTEXT("..RATE..BAD RATE SWITCH ");
        OUTCHAR(TGRP);
        OUTIMAGE;
        RATE :- NONE;
        END;
      END
    END
  END
END ** RATE **;
```

Figure B-173

e. Procedure COUNCK.

i. Purpose. This procedure first checks if the single character country identifier is acceptable (global variable VALIDCOUNTRIES), and if it is the numeric index of the country is returned as the function's value.

ii. Procedure COUNCK Skeleton.

Parameters and Local Variables for Procedure COUNCK.

PROCEDURE: COUNCK		TYPE: integer
PARAMETERS:		
C	text	country code checked/converted
LOCAL VARIABLE:		
I	integer	loop index variable
LOCAL PROCEDURE: none		

Figure B-174

iii. Listing.

Listing of Procedure COUNCK.

```

INTEGER PROCEDURE COUNCK(C);TEXT C;
  BEGIN INTEGER I;
    FOR I:= 1 STEP 1 UNTIL NUMCOUNTRIES DO
      IF C.SUB(1,1)=VALIDCOUNTRIES.SUB(I,1) THEN COUNCK := I;
    END ** COUNCK **;
```

Figure B-175

f. Procedure C.

i. Purpose. This function calls a PRIMOS system routine to obtain elapsed CPU time. It has no effect on the execution of SEAC but is useful to the user as a means to monitor the progress of the simulation in machine terms. (This procedure is probably the only routine that would have to be changed if SEAC were to be converted to another host machine running SIMULA.)

ii. Procedure C Skeleton.

Parameters and Local Variables for Procedure C.

PROCEDURE: C	TYPE: external FORTRAN
PARAMETERS:	
CPUTIME integer	elapsed CPU time
LOCAL VARIABLE: none	
LOCAL PROCEDURE: none	

Figure B-176

iii. Listing.

Listing of Procedure C.

```
EXTERNAL FORTRAN PROCEDURE CTIME ="CTIM$A" IS
  PROCEDURE C (CPUTIME); NAME CPUTIME; INTEGER CPUTIME;;
```

Figure B-177

g. Procedure SVCSCCK.

i. Purpose. This procedure first checks if the single character service identifier is acceptable (global variable VALIDSERVICES), and if it is the numeric index of the service is returned as the function's value.

ii. Procedure SVCSCCK Skeleton.

Parameters and Local Variables for Procedure SVCSCCK.

PROCEDURE: SVCSCCK	TYPE: integer
PARAMETERS:	
C text	service code checked/converted
LOCAL VARIABLE:	
I integer	loop index
LOCAL PROCEDURE: none	

Figure B-178

iii. Listing.

Listing of Procedure SVCSCK.

```
INTEGER PROCEDURE SVCSCK(C);TEXT C;  
  BEGIN INTEGER I;  
    FOR I:= 1 STEP 1 UNTIL VALIDSERVICES.LENGTH DO  
      IF C.SUB(1,1)=VALIDSERVICES.SUB(I,1) THEN SVCSCK := I;  
    END ** SVCSCK **;
```

Figure B-179

h. Procedure DIARY.

i. Purpose. DIARY is a procedure that informs the user how the SEAC execution is progressing. At various stages of data input (eg., creation of the category code table) and the completion of each simulated day a message is written to the MONITOR printfile. These messages can be examined by the user to check model performance and progress. The CPU time since the last message and the total time since execution began are printed.

ii. Procedure DIARY Skeleton.

Parameters and Local Variables for Procedure DIARY.

PROCEDURE: DIARY		TYPE: none
PARAMETERS:		
RELTYM	real	elapsed time since user login
T	text	message to be printed with times
LOCAL VARIABLE: none		
LOCAL PROCEDURE: none		

Figure B-180

iii. Listing.

Listing of Procedure DIARY.

```
PROCEDURE DIARY(RELTYM,T);TEXT T;REAL RELTYM;  
  BEGIN  
    C(CPUT);  
    DELTIME := (.01 * CPUT ) - ELAPSED;  
    ELAPSED := ELAPSED + DELTIME;  
    MONITOR.OUTFIX(RELTYM,3,10);MONITOR.OUTTEXT(T);MONITOR.SETPOS(60);  
    MONITOR.OUTFIX(ELAPSED,2,10);  
    MONITOR.OUTFIX(DELTIME,2,10);MONITOR.OUTIMAGE;  
  END;
```

Figure B-181

i. Procedure GET.

i. Purpose. This procedure looks for an engineer unit in either an installation or region list that has a specified unit type code. If a match is found then GET points to this ENGRUNIT object.

ii. Procedure GET Skeleton.

Parameters and Local Variables for Procedure GET.

PROCEDURE: GET		TYPE: REF(ENGRUNIT)
PARAMETERS:		
EID	text	UTC of engineer unit to find
Q	HEAD	set of ENGRUNITs
LOCAL VARIABLE:		
E	ENGRUNIT	points to objects in the set
LOCAL PROCEDURE: none		

Figure B-182

iii. Listing.

Listing of Procedure GET.

```
REF(ENGRUNIT) PROCEDURE GET(EID,Q); REF(HEAD)Q;TEXT EID;
BEGIN
  REF(ENGRUNIT)E;
  E:-Q.FIRST;
  WHILE E /= NONE DO
    IF E.UTCMSTR.EUTC = EID THEN
      BEGIN GET :- E; E :- NONE; END
    ELSE
      E :- E.SUC;
  END -- GET --;
```

Figure B-183

j. Procedure TYPEINSTALL.

i. Purpose. Procedure TYPEINSTALL is called from BUILDMAP and is used to create an INSTALLATION or subINSTALLATION object of the type indicated on the GEOLOC file. The default CLASS is INSTALLATION.

ii. Procedure TYPEINSTALL Skeleton.

Parameters and Local Variables for Procedure TYPEINSTALL.

PROCEDURE: TYPEINSTALL		TYPE: REF(INSTALLATION)
PARAMETERS:		
IC	integer	country index
ISX	integer	service index
N	text	name of the installation
TYPE	text	installation type code
LOCAL VARIABLE: none		
LOCAL PROCEDURE: none		

Figure B-184

iii. Listing.

Listing of Procedure TYPEINSTALL.

```

REF(INSTALLATION) PROCEDURE TYPEINSTALL(IC,ISX,N,TYPE);
TEXT TYPE,N;INTEGER IC,ISX;
BEGIN
IF IC*ISX <> 0 THEN
  TYPEINSTALL :- IF TYPE = "AFLD" OR TYPE = "COAB" THEN
    NEW AIRBASE(IC,ISX,N)
  ELSE IF TYPE = "APOD" THEN NEW AIRPOD(IC,ISX,N)
  ELSE IF TYPE = "ADA " THEN NEW ADA(IC,ISX,N)
  ELSE IF TYPE = "HOSP" THEN NEW HOSPITAL(IC,ISX,N)
  ELSE IF TYPE = "CAMP" THEN NEW CAMP(IC,ISX,N)
  ELSE IF TYPE = "RS&D" THEN NEW CENTER(IC,ISX,N)
  ELSE IF TYPE = "PORT" THEN NEW PORT(IC,ISX,N)
  ELSE IF TYPE = "SPOD" THEN NEW SEAPOD(IC,ISX,N)
  ELSE IF TYPE = "DEPO" THEN NEW DEPOT(IC,ISX,N)
  ELSE IF TYPE = "ELEC" THEN NEW ELEC(IC,ISX,N)
  ELSE NEW INSTALLATION(IC,ISX,N)
ELSE
  BEGIN SETPOS(10);
  OUTTEXT("..INST.. ERROR IN TYPING INSTALLATION(CO,SVC) ");
  OUTINT(IC,5);OUTINT(ISX,5);OUTTEXT(N);OUTIMAGE;
  END;
END -- TYPEINSTALL --;
```

Figure B-185

k. Procedure DEPLOY.

i. Purpose. One of the particular objectives of SEAC was to improve the way that units destined for the FCZ are treated as they pass through the COMMZ. This is discussed in the main paper under RS&D. Implicit in developing such logic, however, is the need to establish probable unit destinations. In the initial application of SEAC, the force requirements number (FRQN) was judged to be the best indicator for FCZ bound units. DEPLOY and RS&D key on the FRQN and compare its first character to a list of characters that indicate units which probably deploy to the FCZ. Future applications of SEAC might require changes if an alternative to FRQN proves better.

ii. Procedure DEPLOY Skeleton.

Parameters and Local Variables for Procedure DEPLOY.

PROCEDURE: DEPLOY		TYPE: boolean
PARAMETERS:		
FRQN	text	force requirements number from unit arrivals file
LOCAL VARIABLE:		
I	integer	loop variable
LOCAL PROCEDURE: none		

Figure B-186

iii. Listing.

Listing of Procedure DEPLOY.

```
BOOLEAN PROCEDURE DEPLOY(FRQN);TEXT FRQN;  
  BEGIN INTEGER I;  
  FOR I := 1 STEP 1 UNTIL COMBATIDS.LENGTH DO  
    IF COMBATIDS.SUB(1,1) = FRQN.SUB(1,1) THEN DEPLOY := TRUE;  
  END -- DEPLOY --;
```

Figure B-187

1. Procedure BLAST.

i. Purpose. This procedure is used to generate damage to a facility that results from a given type of ordnance. SEAC explicitly generates damage--relating actual threat to probable levels of damage. The threat portion of SEAC works to come up with a possible array of attacks by air and ground forces against COMMZ facilities. BLAST evaluates the attacks and

determines the amount of relative damage that could occur. The main paper discusses the war damage methodology in more detail.

ii. Procedure BLAST Skeleton.

Parameters and Local Variables for Procedure BLAST.

PROCEDURE: BLAST		TYPE: real
PARAMETERS:		
DCRIT	text	damage criteria (EMDA, EM
SIZ	real	size of facility target
TGTSTD	character	construction standard 'I' or 'T'
OTYP	ORDNANCE	type of explosive used
OUNITS	real	amount of ordnance (number or lbs)
LOCAL VARIABLE:		
SUMDAM	real	accumulated damage from attack
DAMAGE	real	damage from current ordnance
WTOEDG	real	distance of blast center to target edge
WANGLE	real	azimuth of blast v. target
BANGLE	real	not used **
CHORD	real	not used **
REFDIST	real	distance from blast center to a target with damage chord passing through target midpoint
REFANGLE	real	central angle for damage arc
REFSEC	real	area of sector damage
MISDIST	real	miss distance
OFFSET	real	distance between target and blast centers
WR	real	weapon radius (blast or fragment)
TGTRAD	real	target radius
TGTAREA	real	target area (n/a for linear target)
HITRNG	real	max offset for damage to occur
BX	real	x coordinate of ordnance center
BY	real	y coordinate of ordnance center
LOCAL PROCEDURE: none		

Figure B-188

iii. Listing.

Listing of Procedure BLAST.

```
REAL PROCEDURE BLAST(DCRIT,SIZ,TGTSTD,OTYP,OUNITS);
  TEXT DCRIT;REAL SIZ,OUNITS; REF(ORDNANCE) OTYP;CHARACTER TGTSTD;
  BEGIN
    REAL SUMDAM,DAMAGE,WTOEDG,WANGLE,BANGLE,CHORD,REFDIST,REFANGLE,REFSEC;
    REAL MISDIST,OFFSET,WR,TGTRAD,TGTAREA,HITRNG,BX,BY;
    IF ATKLOG THEN BEGIN OUTTEXT(" .BLAST. ");OUTTEXT(DCRIT); END;
    IF OTYP IS ROCKET THEN GO TO RETURN;
    TGTRAD := IF DCRIT = "EMDL" THEN (.5 * SIZ) ELSE (.5 * SQRT(SIZ));
    IF OTYP IS BOMB THEN
      BEGIN
        WR := OTYP QUA BOMB.EFFECT(TGTSTD);
        HITRNG := TGTRAD + OTYP QUA BOMB.RFRAG;
        MISDIST := IF DCRIT = "EMDL" OR DCRIT = "EMDA" THEN 0.0 ELSE WR;
      END
    ELSE
      IF OTYP IS DEMOL THEN
        BEGIN
          WR := OTYP QUA DEMOL.EFFECT(TGTSTD,OUNITS);
          HITRNG := TGTRAD;
          MISDIST := 0.0;
        END
        ELSE MISDIST := -2 * TGTRAD;

    WHILE OUNITS > 0.0 DO
      BEGIN
        IF OTYP IS BOMB THEN
          BEGIN
            IF DRAW(OTYP QUA BOMB.DUDRATE,UDUD) THEN BX:=10*HITRNG
              ELSE
                BEGIN
                  BX := UNIFORM(-HITRNG,HITRNG,UDAM);
                  BY:=IF DCRIT = "EMDL" THEN 0.0 ELSE
                    UNIFORM(-HITRNG,HITRNG,UDAM);
                  OFFSET := SQRT( BX*BX + BY*BY );
                END;
            OUNITS := OUNITS - 1.0;
          END
          ELSE
            BEGIN OFFSET:=BX:=UNIFORM(0.0,HITRNG,UDAM);OUNITS:=0.0;END;
        IF ATKLOG THEN
          BEGIN
            OUTTEXT(" DIST:");OUTFIX(OFFSET,2,6);
            OUTTEXT(", RAD:");OUTFIX(TGTRAD,2,6);
          END;
        IF OFFSET > TGTRAD + MISDIST THEN DAMAGE := 0.0
          ELSE
            IF DCRIT = "EMDA" THEN DAMAGE := 1.0
              ELSE
                IF DCRIT = "EMDL" THEN DAMAGE:=WR
                  ELSE

```

```

IF TGTRAD > WR THEN
  BEGIN
    IF OFFSET > TGTRAD THEN
      BEGIN
        WTOEDG := OFFSET - TGTRAD;
        WANGLE := 2 * ARCCOS( WTOEDG/WR);
        DAMAGE := (.5*WR*WR)*( PI*WANGLE/R180 - SIN(WANGLE) );
      END
    ELSE
      BEGIN
        WTOEDG := TGTRAD - OFFSET;
        IF WTOEDG > WR THEN DAMAGE := PI * WR * WR
        ELSE
          BEGIN
            WANGLE := R360 - 2 * ARCCOS( WTOEDG/WR );
            DAMAGE := (.5*WR*WR)*( PI*WANGLE/R180 -
                                SIN(WANGLE));
          END;
        END;
      END
    ELSE
      IF OFFSET + TGTRAD < WR THEN DAMAGE := TGTAREA
      ELSE
        BEGIN
          REFANGLE := 2*ARCSIN(TGTRAD/WR);
          REFDIST := WR * COS(REFANGLE);
          IF OFFSET - TGTRAD > REFDIST THEN
            BEGIN
              WTOEDG := OFFSET - TGTRAD;
              WANGLE := 2 * ARCCOS( WTOEDG/WR);
              DAMAGE := (.5*WR*WR)*(PI*WANGLE/R180-SIN(WANGLE));
            END
          ELSE
            BEGIN
              REFSEC := (.5*WR*WR)*
                (PI*REFANGLE/R180-SIN(REFANGLE));
              DAMAGE:=REFSEC+(TGTRAD+TGTRAD)*
                (TGTRAD-OFFSET+REFDIST);
            END;
          END;
        END;
        SUMDAM := SUMDAM + DAMAGE;
        DAMAGE := 0.0
      END;
    BLAST := SUMDAM;
    RETURN;
  END -- BLAST --;

```

Figure B-189

9. Procedures -- Major. The following procedures, although equivalent to the preceeding functions in format, are much larger in purpose or operation. They are termed "major" procedures to distinguish the more expansive role that they play within SEAC.

a. Procedure BUILDMAP.

i. Purpose. This procedure creates the abstract COMMZ representation. (The main paper and individual CLASS descriptions discuss the region, mapcell, and installation framework used in SEAC.) Two files are read: the region/mapcell file and the GEOLOC file. Definitions evolve in a strict top down fashion: first the regional hierarchy is defined for a country; then mapcells are "mapped" to the (sub)regions that contain them; and finally the GEOLOC data defining COMMZ installations and indicating which mapcell is processed. Inconsistent, missing, or uninterpretable records are reported.

ii. Procedure BUILDMAP Skeleton.

Parameters and Local Variables for Procedure BUILDMAP.

PROCEDURE: BUILDMAP		TYPE: none
PARAMETERS:		
ECHO	boolean	if true then input printed as read
MAPFILE	text	name of region/mapcell def file
GEOFILE	text	name of GEOLOC file
LOCAL VARIABLE:		
TMPR	REGION[]	used to construct region heirarchy
MAPDATA	INFILE	object for MAPFILE
CELLOC	INFILE	object for GEOLOC
CKGEO	GEOLOC	used to determine if GEOLOC exists
GOBJ	GEOLOC	used for new GEOLOC
TMPINSTL	INSTALLATION	used for new INSTALLATION
CIN	INSTALLATION	used for finding INSTALLATIONS
RBUF	text	buffer for region/mapcell input
DEFTYP	character	if 'R' then REGION def else MAPCELL
CBUF	text	BUFFER FOR GEOLOC (INSTALLATIONS)
MC	text	CBUF mapcell index field
GLOC	text	CBUF GEOLOC field
CCOUN	text	CBUF country field
CSVC	text	CBUF service field
CNAM	text	CBUF GEOLOC/INSTALLATION name
CINTYP	text	CBUF installation type
CMAJ	text	CBUF owning installation indicator
RX	integer	numeric index of current region
RXR	integer	numeric index of subordinate region
RXC	integer	numeric index of subordinate mapcel
RECOUN	integer	country on input record
CURCOUN	integer	country being defined (records)
I	integer	loop index
IMC	integer	numeric equivalent of MC
ICOUN	integer	numeric equivalent of CCOUN
ISVC	integer	numeric equivalent of CSVC
RSPRD	integer	country index having road respon.
RSPRR	integer	country index having railroad resp.
RSPPP	integer	country index having pipeline resp
LOCAL PROCEDURE:		
ERROR	none	prints error messages

Figure B-190

iii. Listing.

Listing of Procedure BUILDMAP.

```
PROCEDURE BUILDMAP(ECHO,MAPFILE,GEOFILE);
  TEXT MAPFILE,GEOFILE;
  BOOLEAN ECHO;
  BEGIN
    REF(REGION) ARRAY TMPR(1:100);
    REF(INFILE) MAPDATA,CELLOCS;
    REF(GEOLOC)CKGEO,GOBJ;REF(INSTALLATION)TMPINSTL,CIN;
    TEXT RBUF;CHARACTER DEFTYP;
    TEXT CBUF,MC,GLOC,CCOUN,CSVC,CNAM,CINTYP,CMAJ;
    INTEGER RX,RXR,RXC,RECOUN,CURCOUN,I;
    INTEGER IMC,ICOUN,ISVC,RSPRD,RSPRR,RSPPP;

    PROCEDURE ERROR(T1,T2);TEXT T1,T2;
      BEGIN SETPOS(10);
        OUTTEXT(" ### BUILDMAP ERROR...");
        OUTTEXT(T1);OUTTEXT(T2);
        OUTIMAGE;
        END;

    RBUF :- BLANKS(80);
    MAPDATA :- NEW INFILE(MAPFILE);
    MAPDATA.OPEN(RBUF);

    MAPDATA.INIMAGE;
    CURCOUN := COUNCK(RBUF.SUB(1,1));
    OUTTEXT("..MAPBLD..COUNTRY..");OUTINT(CURCOUN,3);OUTIMAGE;
    TMPR(1) :- RGNSTRUC(1) :- NEW REGION(1,CURCOUN);
    WHILE NOT MAPDATA.ENDFILE DO
      BEGIN
        MAPDATA.IMAGE :- MAPDATA.IMAGE.STRIP;
        IF ECHO THEN BEGIN OUTTEXT(">RGN:");OUTTEXT(RBUF);OUTIMAGE;END;
        RECOUN := COUNCK(MAPDATA.INTXT(1));
        DEFTYP := MAPDATA.INCHAR;
        IF RECOUN NE CURCOUN THEN
          BEGIN
            OUTTEXT("..MAPBLD..COUNTRY CHANGE..");
            OUTINT(RECOUN,3);OUTIMAGE;
            FOR I := 1 STEP 1 UNTIL 100 DO TMPR(I) :- NONE;
            CURCOUN := RECOUN;
            TMPR(1) :- RGNSTRUC(CURCOUN) :- NEW REGION(1,CURCOUN);
            END;
        RX := MAPDATA.ININT;
        IF TMPR(RX) /= NONE THEN
          BEGIN
            IF DEFTYP = 'R' THEN
              BEGIN
                WHILE MAPDATA.MORE DO
                  BEGIN
                    RXR := MAPDATA.ININT;
                    TMPR(RXR) :- NEW REGION(RXR,CURCOUN);
```

```

        TMPR(RXR).INTO(TMPR(RX).SUBRGNS);
        TMPR(RXR).SUPRGN :- TMPR(RX);
        END;
    END
    ELSE
    IF DEFTYP = 'C' THEN
        BEGIN
            RSPRD:=COUNCK(MAPDATA.INTEXT(1));
            RSPRR:=COUNCK(MAPDATA.INTEXT(1));
            RSPPP:=COUNCK(MAPDATA.INTEXT(1));
            WHILE MAPDATA.MORE DO
                BEGIN
                    RXC :- MAPDATA.ININT;
                    IF AREAS(RXC) == NONE THEN
                        AREAS(RXC) :- NEW MAPCELL(RSPRD,RSPRR,
                                                    RSPPP);
                    NEW REGCELL(AREAS(RXC)).
                        INTO(TMPR(RX).SUBRGNS);
                    AREAS(RXC).RGNLINK(CURCOUN) :- TMPR(RX);
                END;
            END
        ELSE
            ERROR("<< WRONG CARDTYPE >> ",RBUF);
        END
    ELSE
        ERROR("<< REGION DOESN'T EXIST >> ",RBUF);
    MAPDATA.IMAGE :- RBUF;
    MAPDATA.INIMAGE;
    END;

MAPDATA.CLOSE;
OUTTEXT("      I. REGION-MAPCELL MAPPING COMPLETE");OUTIMAGE;

CBUF :- BLANKS(80);
CELLOCS :- NEW INFILE(GEOFILE);
CELLOCS.OPEN(CBUF);
CELLOCS.INIMAGE;

MC :- CBUF.SUB(1,3);          GLOC :- CBUF.SUB(5,4);
CCOUN :- CBUF.SUB(10,1);      CSVC :- CBUF.SUB(13,1);
CNAM :- CBUF.SUB(15,20);      CINTYP :- CBUF.SUB(40,4);
CMAJ :- CBUF.SUB(46,4);

WHILE NOT CELLOCS.ENDFILE DO
    BEGIN
        IF ECHO THEN BEGIN OUTTEXT(">GEO:");OUTTEXT(CBUF);OUTIMAGE;END;
        IMC :- MC.SUB(1,3).GETINT;
        IF AREAS(IMC) /= NONE THEN
            BEGIN
                ICOUN := COUNCK(CCOUN.SUB(1,1));
                ISVC := SVCCK(CSVC.SUB(1,1));
                IF (ICOUN * ISVC) > 0 THEN
                    BEGIN
                        CKGEO :- GEOTAB.FIND(GLOC);

```

```

IF CKGEO == NONE THEN
  BEGIN
    IF CINTYP = "COAB" THEN
      ERROR("<< COLOCATED BASE DOESN'T EXIST >> ",
        CBUF)
    ELSE
      IF CINTYP = "CMPX" THEN
        BEGIN
          CKGEO :- GEOTAB.FIND(CMAJ);
          CIN :- MATCH(CKGEO, ICOUN, ISVC);
          IF CIN == NONE THEN
            ERROR("<< COMPLEXING >>", CBUF)
          ELSE
            NEW GEOSUB(GLOC, CKGEO).
              RANK(GEOTAB);
        END
      ELSE
        BEGIN
          GOBJ :- NEW GEOLoc(GLOC);
          GOBJ.RANK(GEOTAB);
          GOBJ.INTO(AREAS(IMC).GEOSSET);
          TMPINSTL :- TYPEINSTALL(ICOUN, ISVC,
            CNAM.SUB(1,20).STRIP,CINTYP);
          TMPINSTL.INTO(GOBJ.SITES);
          IF CINTYP = "AFLD" OR
            CINTYP = "APOD" THEN
            TMPINSTL.DETRQMT("BASE",1.0);
          INSPECT TMPINSTL
            WHEN AIRPOD DO
              GEOCTRS(CBUF.SUB(51,20))
            WHEN SEAPOD DO
              GEOCTRS(CBUF.SUB(51,20))
            WHEN ADA DO DETRQMT("ADA",1.0);
          GOBJ.CELL :- AREAS(IMC);
          END;
        END
      ELSE
        BEGIN
          CIN :- MATCH(CKGEO, ICOUN, ISVC);
          IF CIN /= NONE THEN
            ERROR("[[ INSTAL AMBIGUITY ]]",
              CBUF.SUB(1,50))
          ELSE
            BEGIN
              TMPINSTL :- TYPEINSTALL(ICOUN, ISVC,
                CNAM.SUB(1,20).STRIP,CINTYP);
              IF CINTYP = "COAB" THEN
                BEGIN
                  CIN:-CKGEO.SITES.FIRST;
                  WHILE CIN /= NONE DO
                    IF CIN IN AIRBASE THEN
                      BEGIN
                        CIN QUA AIRBASE.COLOC:-TMPINSTL;
                        TMPINSTL.INTO(CKGEO.SITES);

```

```

        CIN:-NONE;
        END
        ELSE
        CIN:-CIN.SUC;
        IF TMPINSTL.PRED == NONE THEN
            ERROR("    [[ COLOCATION FAILED ]]",
                CBUF);
        END
        ELSE
        BEGIN
        IF CINTYP = "AFLD" OR
            CINTYP = "APOD" THEN
            TMPINSTL.DETRQMT("BASE",1.0);
        INSPECT TMPINSTL
            WHEN AIRPOD DO
                GEOCTRS(CBUF.SUB(51,20))
            WHEN SEAPOD DO
                GEOCTRS(CBUF.SUB(51,20))
            WHEN ADA DO DETRQMT("ADA",1.0);
        TMPINSTL.INTO(CKGEO.SITES);
        END;
        END;
        END;
        ELSE
        ERROR("<< BAD COUNTRY SERVICE >> ",CBUF);
        END
        ELSE
        ERROR("<< MISSING MAPCELL AREA >> ",CBUF);
        CELLOCS.INIMAGE;
        END;

        CELLOCS.CLOSE;
        OUTTEXT("    II. GEOLOC TREE FINISHED");OUTIMAGE;

        END ** BUILDMAP **;

```

Figure B-191

b. Procedure ADDASSETS.

i. Purpose. This procedure reads the ASSET file and instigates the initial creation and assignment of facility assets to installations and mapcells. In the case of installation assets the target location is identified by its GEOLOC-country-service designation. MSR assets, i.e. roads, railroads, and pipelines, are assigned according to the mapcell index.

ii. Procedure ADDASSETS Skeleton.

Parameters and Local Variables for Procedure ADDASSETS

PROCEDURE: ADDASSETS		TYPE: none
PARAMETERS:		
ASSETFYL	text	name of the ASSET file
LOCAL VARIABLE:		
ITEM	text	input buffer for file
LOKEY	text	ITEM location field (inst or cell)
LSTKEY	text	copy of last LOKEY
CCODE	text	ITEM category code field
SIZE	text	ITEM catcode size field
UOM	text	ITEM catcode unit of meas field
CNTRY	text	ITEM country field
SRVC	text	ITEM service field
ARINDX	integer	numeric equivalent of mapcell LOKEY
RSIZE	real	numeric equivalent of SIZE
PRINTERR	boolean	flags status of present location
G	GEOLOC	object for installation LOKEY
P	PROPERTY	object associated with CCODE type
AFAC	FACILITY	not used **
MPC	MAPCELL	object identified by ARINDX
CTC	CATCOD	not used **
INS	INSTALLATION	object for LOKEY-CNTRY-SRVC
CMPO	COMPO	not used **
LOCAL PROCEDURE: none		

Figure B-192

iii. Listing.

Listing of Procedure ADDASSETS

```

PROCEDURE ADDASSETS(ASSETFYL);
  TEXT ASSETFYL;
  BEGIN
    REF(INFILE) ASSETS;
    TEXT ITEM, LOKEY, LSTKEY, CCODE, SIZE, UOM, CNTRY, SRVC;
    INTEGER ARINDX; REAL RSIZE;      BOOLEAN PRINTERR;
    REF(GEOLOC)G;   REF(PROPERTY)P;   REF(FACILITY)AFAC;
    REF(MAPCELL)MPC; REF(CATCOD)CTC;   REF(INSTALLATION)INS;
    REF(COMPO)CMPO;
    ITEM :- BLANKS(120);
    ASSETS :- NEW INFILE(ASSETFYL);
    ASSETS.OPEN(ITEM);
    ASSETS.INIMAGE;
    LOKEY :- ITEM.SUB(1,4); LSTKEY :- BLANKS(4);
    CCODE :- ITEM.SUB(11,4); SIZE :- ITEM.SUB(21,10);
    UOM :- ITEM.SUB(33,2); CNTRY :- ITEM.SUB(17,1);
    SRVC :- ITEM.SUB(19,1);
    WHILE NOT ASSETS.ENDFILE DO
  
```

```

BEGIN
IF LOKEY <> LSTKEY THEN
  BEGIN
  INS:-NONE; MPC:-NONE;
  IF NOT LETTER(LOKEY.SUB(1,1).GETCHAR) THEN
    BEGIN
    ARINDX := LOKEY.SUB(1,4).GETINT;
    MPC :- AREAS(ARINDX);
    END
  ELSE
  BEGIN
  G:-GEOTAB.FIND(LOKEY);
  INS:-MATCH(G,COUNCK(CNTRY.SUB(1,1)),
              SVCSCK( SRVC.SUB(1,1)));
  END;
  LSTKEY := LOKEY;
  PRINTERR := TRUE;
  END;
CMPO :- NONE;
RSIZE := SIZE.SUB(1,10).GETREAL;
IF INS /= NONE THEN
  BEGIN
  P :- INS.ASSETS.FIND(CCODE);
  IF P == NONE THEN
    BEGIN
    P :- NEW PROPERTY(CCODE);
    IF P.XREF(INS) THEN P.RANK(INS.ASSETS);
    END;
  IF P.CMP /= NONE THEN P.SHOWS(RSIZE,UOM);
  END
  ELSE
  IF MPC == NONE THEN
    BEGIN
    IF PRINTERR THEN
      BEGIN SETPOS(20);
      OUTTEXT("..ASSETS...PLACE ERROR--");
      OUTTEXT(ITEM.SUB(1,40));
      OUTIMAGE;
      PRINTERR := FALSE;
      END;
    END
  ELSE
  IF NOT MPC.HASA(CCODE,COUNCK(CNTRY.SUB(1,1)),
                  SVCSCK(SRVC.SUB(1,1)),UOM,RSIZE,XTSTD) THEN
    BEGIN
    OUTTEXT("..ASSETS..CATCODE NOT ID'D ");
    OUTTEXT(ITEM.SUB(1,40));
    OUTTEXT(", IN AREA ");OUTINT(ARINDX,5);
    OUTIMAGE;
    END;
  
```

```

    ASSETS.INIMAGE;
    END;
ASSETS.CLOSE;
OUTTEXT("      VI. ASSET FILE ENTERED");OUTIMAGE;
END -- ADDASSETS --;

```

Figure B-193

c. Procedure SETUP.

i. Purpose. This procedure defines and initializes the anticipated enemy air assets that will be used against the COMMZ as found on the threat setup file. (See Data File Annex for datum itemization.) The following data is read: facility group target distributions by installation subCLASS; superficial and structural thresholds as well as effective miss distances for facility subCLASSES; distribution of airfield runway crater priorities; notionalized span lengths for masonry and steel bridges; auxiliary damage factors for bridges, rail tracks, pipelines, roads, storage tanks, and storage bladders; entry of ordnance type and characteristics for enemy planes and infiltration units; entry of plane types, characteristics and payloads; listing of enemy airbases; and finally the inventory of planes by enemy airbase at the start of the scenario.

ii. Procedure SETUP Skeleton.

Parameters and Local Variables for Procedure SETUP.

PROCEDURE: SETUP		TYPE: none
PARAMETERS:		
TAIR	text	name of threat information file
ECHO	boolean	flag to print input data
LOCAL VARIABLE:		
START	INFILE	object for TAIR
BASE	NKAIRBASE	enemy airbase object reference
TGP	TGTPREF	object reference
PT	AIRCRAFT	points to objects in PLANETYPES set
P	AIRCRAFT	indicates present AIRCRAFT type
FRAME	AIRCRAFT	used to define AIRCRAFT types
ACGRP	HEAD	indicates which set to place new PLANE (either bombers or fighters)
ORDCLS	ORDNANCE	points to match in ordnance table
SBUF	text	input buffer for START
RECTYPE	text	single character record type ID
INGEO	text	SBUF enemy airbase field
PTYPE	text	SBUF aircraft type field
NOPLANES	text	SBUF number of type planes field
MAXRNG	text	not used **
ACID	text	SBUF base aircraft type field
ACCAT	text	SBUF bomber or fighter id field
ACRNG	text	SBUF aircraft max range field
ACSPEED	text	SBUF aircraft speed field
ORDNCID	text	SBUF ordnance name field
ORDID	text	SBUF ordnance type field
I	integer	loop index
K	integer	loop index
L	integer	loop index
NT	integer	vector dimension for installation target preference
NUMPLANES	integer	numeric equivalent of NOPLANES
LOCAL PROCEDURE: none		

Figure B-194

iii. Listing.

Listing of Procedure SETUP.

```
PROCEDURE SETUP(TAIR,ECHO);
  TEXT TAIR;BOOLEAN ECHO;
  BEGIN
    REF(INFILE)START;   REF(NKAIRBASE)BASE;   REF(TGTPREF) TGP;
    REF(AIRCRAFT)PT,P;   REF(HEAD)ACGRP;       REF(AIRCRAFT)FRAME;
    REF(ORDNANCE)ORDCLS;
    TEXT SBUF, RECTYPE, INCEO, PTYPE, NOPLANES, MAXRNG;
    INTEGER I,K,L,NT,NUMPLANES;
    TEXT ACID,ACCAT,ACRNG,ACSPPEED,ORDNCID,ORDID;

    SBUF :- BLANKS(80);
    START :- NEW INFILE(TAIR);
    START.OPEN(SBUF);
    START.INIMAGE;

    PKILL:-START.INREAL;   UKILL:-START.ININT;   START.INIMAGE;
    PSUC:-START.INREAL;    USUC:-START.ININT;    START.INIMAGE;
    PBEFOR:-START.INREAL;  UBEFOR:-START.ININT;   START.INIMAGE;
    PDAM:-START.INREAL;    UDAM:-START.ININT;    START.INIMAGE;
    PHIT:-START.INREAL;    UHIT:-START.ININT;    START.INIMAGE;
    XREP:-START.INREAL;    UREP:-START.ININT;    START.INIMAGE;
    RELRATE:-START.INREAL;                                START.INIMAGE;
    URDY:-START.ININT;                                         START.INIMAGE;
    IF ECHO THEN BEGIN OUTTEXT("--SETUP RANDOM NUMBERS SET");OUTIMAGE;END;
    FOR K :- 1 STEP 1 UNTIL 8 DO FACHIT(K) :- START.INREAL;
    START.INIMAGE;
    UFHIT :- START.ININT;

    FOR K :- 2 STEP 1 UNTIL XINSTAL DO
      BEGIN
        START.INIMAGE;
        START.SETPOS(11); NT :- START.ININT;
        TGTDEFAULTS(K) :- NEW TGTPREF(NT);
        INSPECT TGTDEFAULTS(K) DO
          FOR L :- 1 STEP 1 UNTIL NT DO VECTOR(L) :- START.INREAL;
        END;
        IF ECHO THEN BEGIN OUTTEXT("--SETUP TARGETING DATA IN");OUTIMAGE;END;
        START.INIMAGE;
        FOR K :- 1 STEP 1 UNTIL XFACS DO
          BEGIN
            SUPRFICL(K,1):-START.INREAL;   SUPRFICL(K,2):-START.INREAL;
            STRUCT(K,1):-START.INREAL;     STRUCT(K,2):-START.INREAL;
            EMDTAB(K):-START.INREAL;        START.INIMAGE;
          END;
          IF ECHO THEN BEGIN OUTTEXT("--SETUP FACILITY VALUES SET");OUTIMAGE;END;
          CRTRSPLIT(1):-START.INREAL;   CRTRSPLIT(2):-START.INREAL;
          CRTRSPLIT(3):-START.INREAL;   START.INIMAGE;
          SPNLEN(1):-START.INREAL;      SPNLEN(2):-START.INREAL;   START.INIMAGE;
          DRPSpan(1):-START.INREAL;     DRPSpan(2):-START.INREAL;   START.INIMAGE;
```

```

BRGFAC := START.INREAL;                                START.INIMAGE;
RRDAMFAC := START.INREAL;    RDDAMFAC := START.INREAL;
TNKDAMFAC := START.INREAL;    PIPDAMFAC := START.INREAL; START.INIMAGE;
HTANK := START.INREAL;        HBLAD := START.INREAL;    START.INIMAGE;

RECTYPE :- SBUF.SUB(1,1);
ORDID :- SBUF.SUB(3,1);    ORDNCID :- SBUF.SUB(6,5);
ORDTAB :- NEW ORDLIST;

WHILE RECTYPE = "W" DO
  BEGIN
    IF ECHO THEN BEGIN OUTTEXT(">MUN:");OUTTEXT(SBUF);OUTIMAGE;END;
    START.SETPOS(11);
    IF ORDID = "B" THEN
      NEW BOMB(ORDNCID,START.INREAL,START.INREAL,START.INREAL).
        INTO(ORDTAB)
    ELSE IF ORDID = "R" THEN
      NEW ROCKET(ORDNCID,START.INREAL).INTO(ORDTAB)
    ELSE IF ORDID = "D" THEN
      NEW DEMOL(ORDNCID,START.INREAL,START.INREAL).
        INTO(ORDTAB)
    ELSE
      IF ECHO THEN BEGIN OUTTEXT("    BAD ORDNANCE");OUTIMAGE;END
      ELSE
        BEGIN
          OUTTEXT(SBUF);OUTTEXT("  HAS AN ERROR");OUTIMAGE;
        END;
    START.INIMAGE;
  END;

ACID :-SBUF.SUB(6,5);    ACCAT :-SBUF.SUB(12,1);
ACRNG :-SBUF.SUB(16,5);    ACSPEED :-SBUF.SUB(21,5);

WHILE RECTYPE = "A" DO
  BEGIN
    IF ECHO THEN BEGIN OUTTEXT(">PLN:");OUTTEXT(SBUF);OUTIMAGE;END;
    FRAME :- IF ACCAT="B" THEN
      NEW BMBR(ACID, ACRNG.SUB(1,5).GETREAL,
        ACSPEED.SUB(1,5).GETREAL)
      ELSE
        NEW FTR(ACID, ACRNG.SUB(1,5).GETREAL,
          ACSPEED.SUB(1,5).GETREAL);
    FRAME.INTO(PLANETYPES);
    START.SETPOS(30);
    WHILE START.MORE DO
      BEGIN
        ORDCLS:-ORDTAB.FIND(START.INTEXT(5));
        IF ORDCLS /= NONE THEN
          BEGIN
            NEW MUNITION(ORDCLS,START.ININT)
              .INTO(FRAME.PAYLOAD);
          END
        ELSE
          BEGIN

```

```

        IF NOT ECHO THEN OUTTEXT(START.IMAGE.SUB(1,START.POS))
        ELSE SETPOS(START.POS);
        OUTTEXT("^ MUNITION NOT FOUND,PROCESSING STOPPED");
        OUTIMAGE;
        START.SETPOS(START.LENGTH + 1);
        END;
    END;
    START.INIMAGE;
    END;

    INGEO :- SBUF.SUB(7,4);
    WHILE RECTYPE = "N" AND NOT START.ENDFILE DO
        BEGIN
            NEW NKAIRBASE(INGEO).INTO(NKAB.FIELDS);
            START.INIMAGE;
            END;
    PTYPE :- SBUF.SUB(12,5);    NOPLANES :- SBUF.SUB(18,3);
    WHILE RECTYPE = "P" AND NOT START.ENDFILE DO
        BEGIN
            IF ECHO THEN BEGIN OUTTEXT(">AOB:");OUTTEXT(SBUF);OUTIMAGE;END;
            BASE :- NKAB.FIND(INGEO);
            IF BASE -/= NONE THEN
                BEGIN
                    P :- NONE;
                    PT:-PLANETYPES.FIRST;
                    WHILE PT -/= NONE DO
                        IF PT.ID = PTYPE THEN
                            BEGIN P :- PT; PT :- NONE; END
                        ELSE
                            PT :- PT.SUC;
                    IF P -/= NONE THEN
                        BEGIN
                            NUMPLANES := NOPLANES.SUB(1,3).GETINT;
                            ACGRP:-IF P IS BMBR THEN BASE.BOMBERS ELSE BASE.FIGHTERS;
                            FOR I :- 1 STEP 1 UNTIL NUMPLANES DO
                                NEW PLANE(P).INTO(ACGRP);
                            END
                        ELSE
                            BEGIN
                                OUTTEXT("..SETUP.. UNRECOGNIZED PLANE TYPE ");
                                OUTTEXT(PTYPE);
                                OUTIMAGE;
                                END;
                            END;
                        END;
                    START.INIMAGE;
                    END;
                START.CLOSE;
                OUTTEXT("      VII.  THREAT SETUP PROCEDURE COMPLETED");
                OUTIMAGE;
                END **SETUP ** ;

```

Figure B-195

d. Procedure WORK.

i. Purpose. This procedure is called once each day to perform outstanding or new work tasks to the extent that engineer capability permits. It looks at each mapcell for MSR related work and each installation that may be in that mapcell for other facility related work. It does this sequentially for each priority level. First mapcells and installations vital tasks are examined, then critical tasks, and so on.

ii. Procedure WORK Skeleton.

Parameters and Local Variables for Procedure WORK.

PROCEDURE: WORK		TYPE: none
PARAMETERS: none		
LOCAL VARIABLE:		
PX	integer	loop index (for each priority)
IMX	integer	loop index (mapcells)
RESPCOUN	integer	country responsible for task
TT	TASK	points to next TASK to be checked
IX	INSTALLATION	loops through SITE sets
GX	GEOLOC	loops through GOSET sets
PIECE	real[]	used to track portion of TASK's horizontal, vertical, & other workload that can be performed
MINPIECE	real	threshold percentum for work to be performed
LOCAL PROCEDURE: none		

Figure B-196

iii. Listing.

Listing of Procedure WORK.

```
PROCEDURE WORK;
  BEGIN
    INTEGER PX,IMX,RESPCOUN;REF(TASK)TT;
    REF(INSTALLATION)IX;REF(GEOLOC)GX;
    REAL ARRAY PIECE(1:3); REAL MINPIECE;

    FOR PX := 1, 2, 3, 4 DO
      FOR IMX := 1 STEP 1 UNTIL NUMCELLS DO
        INSPECT AREAS(IMX) WHEN MAPCELL DO
          BEGIN
            IF PX = 1 THEN TT:-TASKPTR :- TASKLIST.FIRST;
            WHILE TT /= NONE DO
              IF TASKPTR.PRIO <= PX THEN
                BEGIN
                  INSPECT TASKPTR DO
                    BEGIN
                      RESPCOUN := IF FACPTR IS ROAD THEN RDRESP
                        ELSE IF FACPTR IS RR THEN RRESP
                          ELSE IF FACPTR IS PIPE THEN PIPRESP
                            ELSE 0;
                      IF RESPCOUN > 0 THEN
                        BEGIN
                          IF RGNLINK(RESPCOUN).CANDO(WHRZ,WVER,
                            WOTH,PX,PIECE) THEN
                            BEGIN TT :- SUC; PERFORM(1.0); END
                          ELSE
                            BEGIN
                              PIECE(1):=IF WHRZ = 0 THEN 0.0
                                ELSE PIECE(1)/WHRZ;
                              PIECE(2):=IF WVER = 0 THEN 0.0
                                ELSE PIECE(2)/WVER;
                              PIECE(3):=IF WOTH = 0 THEN 0.0
                                ELSE PIECE(3)/WOTH;
                              IF PIECE(1) > PIECE(2) THEN
                                BEGIN
                                  IF PIECE(1) > PIECE(3) THEN
                                    MINPIECE:=1-PIECE(1)
                                  ELSE MINPIECE := 1 - PIECE(3);
                                END
                              ELSE
                                IF PIECE(2) > PIECE(3) THEN
                                  MINPIECE:=1-PIECE(2)
                                ELSE MINPIECE := 1 - PIECE(3);
                              IF MINPIECE >= PIECEWORK THEN
                                BEGIN
                                  IF RGNLINK(RESPCOUN).CANDO(
                                    MINPIECE*WHRZ,MINPIECE*WVER,
                                    MINPIECE*WOTH,PX,PIECE) THEN
                                    BEGIN
                                      TT :- SUC; PERFORM(1.0);
                                      END
                                    ELSE

```

```

                                TT :- SUC;
                                END
                                ELSE
                                TT :- SUC;
                                END;
                                END;
                                END;
                                TASKPTR :- TT;
                                END
                                ELSE
                                TT :- NONE;
                                GX :- GEOSSET.FIRST;
                                WHILE GX /= NONE DO
                                BEGIN
                                IX:-GX.SITES.FIRST;
                                WHILE IX /= NONE DO
                                BEGIN
                                IX.TRIAGE(THIS MAPCELL,PX);
                                IX:-IX.SUC;
                                END;
                                GX:-GX.SUC;
                                END;
                                END;
                                END -- WORK --;

```

Figure B-197

10. CLASS/PROCEDURE Index.

ADA B-147	MAPCELL B-27	TRANSP0 B-113
ADDASSETS B-191	MEDICAL B-105	TYPEINSTALL B-180
ADMIN B-105	MISGEN B-161	UNOFMS B-70
AIRBASE B-138	MISSION B-58	UOMTREE B-49
AIRCRAFT B-85	MUNITION B-85	UTILITY B-109
AIRPOD B-141	NKAIRBASE B-89	WASTE B-112
AUNIT B-115	NKBASES B-31	WATER B-113
AVLTREE B-14	NODE B-104	WORK B-199
BLAST B-182	ORDLIST B-55	
BOMB B-81	ORDNANCE B-80	
BUILDING B-101	PAVEVEMENT B-101	
BUILDMAP B-186	PETRO B-106	
C B-177	PIER B-104	
CAMP B-142	PIPE B-108	
CATCOD B-61	PLANE B-86	
CATOFAC B-172	POLFACIL B-107	
CATREE B-41	POLICY B-33	
CENTER B-143	PORT B-145	
CHANGE B-153	POWER B-112	
COMO B-103	PRIORANK B-174	
COMPO B-69	PROCS B-20	
COMPONENT B-24	PROPERTY B-73	
COUNCK B-177	QTRS B-106	
DATA2 B-17	RANDPICK B-171	
DATAI B-15	RATE B-175	
DEMOL B-83	REARAXN B-165	
DEPLOY B-182	REGCELL B-126	
DEPOT B-148	REGION B-116	
DIARY B-179	REPORTER B-34	
ECAPTREE B-47	REPORTS B-36	
ELEC B-148	REVTMENT B-104	
ENGRCAP B-67	ROAD B-115	
ENGRPOOL B-25	ROCKET B-82	
ENGRUNIT B-78	ROOT B-39	
FACELEM B-27	RR B-115	
FACILITY B-95	RSD B-149	
FACTOR B-63	RUNWAY B-101	
FACTORX B-65	SEAKM B-22	
FACTREE B-43	SEAPOD B-146	
GEOLOC B-70	SETUP B-194	
GEOSUB B-72	SHOP B-104	
GEOTREE B-50	SPFUNIT B-91	
GET B-180	STORAGE B-105	
HARDSTAND B-101	SURFACE B-99	
HEAD B-54	SVCSCCK B-178	
HITS B-127	TANK B-107	
HOSPITAL B-148	TASK B-128	
INSTALLATION B-131	TEAMORG B-92	
LINK B-56	TGTPREF B-52	
LINKAGE B-53	THREAT B-21	

ANNEX C

DESCRIPTION OF SEAC'S DATA FILES

ANNEX C

DESCRIPTION OF SEAC'S DATA FILES

<u>Paragraph</u>		<u>Page</u>
1	Purpose	C-2
2	Approach	C-2
3	Data	C-2
4	Facility Planning Factor Data File	C-2
5	Facility Construction and Repair Data File	C-3
6	Asset Data File	C-4
7	Force List Data File	C-5
8	Engineer Unit Capability Data File	C-6
9	Region Data File	C-6
10	GEOLOC Data File	C-7
11	Air Threat Data File	C-8
12	SOF Threat Data File	C-9
13	Orders File	C-11
14	Priority/Policy File	C-12

Figure

C-1	Data Format for Facility Planning Factor File	C-3
C-2	Data Format for CATCODE File	C-4
C-3	Data Format for Asset File	C-5
C-4	Data Format for the TROOP File	C-6
C-5	Data Format for Engineer Capability File	C-7
C-6	Data Format for Region File	C-8
C-7	Data Format for GEOLOC File	C-9
C-8	Data Format for the Air Threat File	C-10
C-9	Data Format for the SOF Threat	C-10
C-10	Data Format for the Orders File	C-12
C-11	Data Format for Priority/Policy File	C-13

1. Purpose. This annex describes the major data files used by ESC's SEAC model.

2. Approach. This annex:

- a. Describes the nature and source of each data type used in SEAC.
- b. Describes the format of the file as it is stored on the computer and listed in the appendices to this annex.
- c. Indicates the normal security level of the file.
- d. Provides comments about that the file contains, and how it is developed if necessary.

3. Data. The data described in this annex comprises with the input data used by SEAC. One of the major sources of data for SEAC was the CESP. Copies of CESP files must be obtained from Joint Chiefs of Staff (JCS). (Manipulating the data is largely done by using a full screen editor since the data are accessed from disk files.)

4. Facility Planning Factor Data File. Engineer workload is a function of the expected numbers and types of facilities that are necessary to support the force. Facility planning factors are used by planners to determine what type of typical support facilities are associated with a forward-deployed or deploying unit.

a. Source. The principal sources of planning factors used by SEAC are the CESP's PLNGFACT (planning factor) and MASTER Files. The PLNGFACT File is based on data and guidance found in the JCS Memorandum 201-81.¹ The MASTER File contains information on unit facility requirements that are compiled by the individual US services.

b. Content and format. Figure C-1 displays the content and format of the SEAC facility PLNGFACT file.

c. Classification. Much of the data included in the Factor File are classified. In particular, unit and equipment facility needs can be SECRET.

d. Comments. The Factor File was designed to handle all factors used by SEAC. ESC made additions to the file to either enter minimum installation facility requirements for classes of installations, or to reflect

¹Planning Factors for Military Construction in Contingency Operations, Enclosure to Joint Chiefs of Staff Memorandum (MJCS) 201-81 (JCS, 31 October 1981).

DATA FORMAT FOR FACILITY PLANNING FACTOR FILE

Data Item	Start Column	Length	Type	Comments
Factor-key	1	6	Text	UTC, equipment item installation code, etc.
Nation*	8	1	Text	For example, 'A':= us
Service*	9	1	Text	--
Unit of Measure*	18	2	Text	'X' := continuation
Rate	21	10	Real	

*If nation and service columns are blank, then the factor applies to all data items.

Figure C-1

the reduced support requirements mandated for troops moving through the RS&D pipeline.

5. Facility Construction and Repair Data File. While the various planning factors dictate the type and amount of facilities that are needed, additional data are needed to determine the engineer labor requirement associated with constructing each facility. The labor estimate must also indicate how much of the effort will be in horizontal, vertical, and other (general) engineer skill categories. The CATCODE File furnishes these data. The number of the file is derived from the JCS facility category code for military facilities which is the basic facility class identifier used in the file.

a. **Source.** The CESPG Facility Component Definition (CMPNT) File is the primary source of data, but Army Technical Manual 5-301-1 concerning the Army Facilities Components System (AFCS) provides other information that supplements the CMPNT FILE.²

b. **Content and format.** Figure C-2 displays the content and format of the SEAC CATCODE File.

c. **Classification.** The CATCODE File is UNCLASSIFIED.

²Installation Planning Temperate Zone, Army Technical Manual (TM) 5-301-1 (HQDA, August 1981).

DATA FORMAT FOR CATCODE FILE

Data Item	Start Column	Length	Type	Comments
Nation	9	1	Text	--
Service	10	1	Text	--
Facility name	12	18	Text	--
Component size	31	6	Real	--
Unit of Measure	38	2	Text	--
Partition	41	1	Text	--
Horizontal	42	7	Integer	Daily manhour requirement
Vertical	49	7	Integer	Daily manhour requirement
Other	56	7	Integer	Daily manhour requirement
Minimum days	64	3	Integer	Minimum days to complete
Facility code	68	4	Text	CATCODE
Component type	73	1	Text	E = Initial; C = temporary; W = Repair

Figure C-2

d. **Comments.** ESC relied, to a great degree, on the CESPG's CMPNT File, but there was a significant shift in the underlying philosophy in how the file was used. The CMPNT File might have several components listed for each class of work: beddown, restoration, construction, or damage repair. ESC instead includes at most three components: an initial construction standard; a temporary construction standard; and a standard for damage repair. The damage repair component applies only to facilities of a temporary standard. Damage to emergency or initial standard facilities will be replaced by similar structures.

6. **Asset Data File.** The Asset File is the list of facilities located in the COMMZ that can be used to satisfy facility requirements. This list includes facilities at US or foreign installations identified by their geographical location code (GEOLOC), as well as an estimate of MSR road, railroad, and pipeline assets. Host nation (HN) supplied facilities are also included.

a. **Source.** The CESPG ASSET and HNASET Files are the primary source of asset data. Where facilities at known installations seem low or are missing, the various service real property inventory files should be consulted. The Integrated Facility System (IFS) Data Base, for example,

maintained by the US Army Engineering and Housing Support Center (EHSC), can be used to supplement US Army installation data.

b. **Content for format.** Figure C-3 displays the content and format of the SEAC Asset File.

DATA FORMAT FOR ASSET FILE

Data Item	Start Column	Length	Type	Comments
Location	1	4	Text	GEOLOC or mapcell number
Nation	17	1	Text	--
Service	19	1	Text	--
Facility	11	4	Text	CATCODE
Size	21	10	Integer	--
Unit of Measure	33	2	Text	--

Figure C-3

c. **Classification.** Since HN-supplied assets are usually considered to be classified information, the ASSET File is classified if such data is present.

d. **Comments.** ESC found that it is not uncommon for a number of GEOLOCs (i.e., installations) to be absent from the CESP ASSET File. While the IFS data helped to fill in the gaps, ESC had more than a little difficulty translating the Army location (ARLOC) codes on the IFS into GEOLOCs. In addition to having to map the IFS's ARLOCs to GEOLOCs, it is also necessary to translate IFS's AFCS facility codes into JCS CATCODE equivalents. While doing this, ESC also considered the possible recycling of some peacetime facilities. For example, AFCS codes indicating family housing were converted into troop housing assets for the purposes of this study.

7. **Force List Data File.** The Force List File used by SEAC is derived from the TPFDD that governs the OPLAN.

a. **Source.** The CESP TROOP file was used as the primary source of the Force List File used by SEAC. This file is an extract of the TPFDD, edited to eliminate missing or erroneous data. There is no reason, however, why the force list may not be extracted directly from the TPFDD, other than the need to "scrub" the data.

b. **Content and format.** Figure C-4 displays the content and format of the SEAC TROOP File.

DATA FORMAT FOR THE TROOP FILE

Data Item	Start Column	Length	Type	Comments
Force requirement number	7	5	Text	FCZ-bound units
UTC	19	5	Text	UTC
Destination	24	4	Text	GEOLOC
POD	37	4	Text	GEOLOC
Time of debarkation	41	31	Integer	--
Service	47	1	Text	--
Strength	31	6	Integer	--

Figure C-4

c. **Classification.** The data in the TROOP File is classified, usually at the SECRET level.

8. Engineer Unit Capability Data File. Daily engineer capability in SEAC is determined by estimating the available engineer manhours, broken down by horizontal, vertical, and general skills, as found in the engineer units supporting the COMMZ.

a. **Source.** The CESPGE Engineer Unit Capability File (ECAPB) was the source of daily available skill hours for US engineer units. ESC also used entries on the ECAPB to assign a manhour value to the support expected from foreign civilian contractors. Information on foreign military and foreign and US civilian engineer capabilities can also be included if such units have a role in the scenario. Obtaining data on skill breakdowns for such organizations, however, can be difficult if manpower and equipment composition is sketchy.

b. **Content and format.** Figure C-5 displays the content and format of the SEAC Engineer Unit Capability File.

c. **Classification.** The SEAC Engineer Unit Capability File is normally UNCLASSIFIED.

9. Region Data File. The region is a device used by the SEAC model to allocate engineer support on an area basis. The easiest way to visualize it

DATA FORMAT FOR ENGINEER CAPABILITY FILE

Data Item	Start	Length	Type
	Column		
UTC	2	5	Text
Nation	8	1	Text
Service	10	1	Text
Strength	28	4	Real
Horizontal	33	5	Real
Vertical	38	5	Real
Other	43	5	Real

Figure C-5

is to think of a governmental hierarchy. A country has a number of states or provinces; each of these may have a number of counties; each county has a number of cities. Regions are similarly defined, but based on a hierarchy that is a function of planned engineer areas of responsibilities. (The region was designed to remedy the limitation of the base complexing structure found in the CESP. For more information on regions, see Annex D, User Guidance.)

a. **Source.** The regions are defined based on reviews of the OPLAN and concepts of engineer operation. If these plans are well formulated, it should be relatively easy to structure regions. If ill-defined, then subjective judgment must be used.

b. **Content and format.** Figure C-6 displays the content and format of the SEAC Region File.

c. **Classification.** The SEAC Region File is UNCLASSIFIED.

d. **Comments.** The regional concept is one that works well within SEAC, but may initially be difficult to understand. Engineer units can be assigned to either an installation or a region. For those engineer units not assigned to an installation, the region defines the area (and therefore the installations) that the unit can support. Whereas regions may and most often do encompass subregions, mapcells do not form such hierarchy. The mapcell is a unique partition of the COMMZ, and links installations to regions.

10. GEOLOC Data File.

a. **Source.** A GEOLOC is a four-character code for all types of locations (civilian and military) throughout the world within SEAC; GEOLOCs

DATA FORMAT FOR REGION FILE

Data Item	Start Column	Length	Type	Comments
Nation	1	1	Text	--
Record type	2	1	Text	'R' = region; 'C' = mapcell
Region record				--
Region (super)	?	*	Integer	Region being defined
Region(s) (sub)	?	*	Integer	Region's subregion(s)
Mapcell record				--
Region	?	*	Integer	Region being defined
MSR responsibility	?	3	Text	Road, railroad, pipe- line
Mapcell(s)	?	*	Integer	Region's mapcell(s)

?: = free format.

*: = no fixed length.

Figure C-6

identify installations and cities. It is assigned by JCS. There is no single source for GEOLOCs, but the starting point is generally a CESPGE GEOLOC File, if one exists for the OPLAN understudy.

b. **Content and format.** Figure C-7 displays the content and format of the SEAC GEOLOC File.

c. **Classification.** The GEOLOC File is UNCLASSIFIED.

d. **Comments.** The CESPGE GEOLOC File also defines to which base complex a GEOLOC belongs. Instead of mapping GEOLOC to a base complex, SEAC assigns GEOLOCs to SEAC mapcells. In addition, GEOLOC-identified installations are categorized by purpose. Ports, camps, airbases, and depots are some of the categories. Furthermore, if the GEOLOC is a POD, a list of reception centers (by service) can be included. (Annex D and the Main Paper elaborate on the significance of installation type.) Also, if several GEOLOCs are proximate, they can be mapped into one GEOLOC, using the Major field to identify it.

11. **Air Threat Data File.** The Air Threat File defines a scenario of enemy sorties directed against COMMZ targets.

DATA FORMAT FOR GEOLOC FILE

Data Item	Start Column	Length	Type	Comments
Mapcell	1	3	Integer	--
GEOLOC	5	4	Text	--
Nation	10	4	Text	--
Service	13	4	Text	--
Name	15	20	Text	--
Type	40	4	Text	Installation class
Major	44	4	Text	Superior GEOLOC
Reception	?	*	Text	Reception center list

?: = free format.

*: = no fixed length.

Figure C-7

a. **Source.** The sources used to develop missions and sorties will be whatever intelligence or planning documents exist for the scenario.

b. **Content and format.** Figure C-8 displays the content and format of the SEAC Air Threat File.

c. **Classification.** A sample listing of the Air Threat File will be determined from the source materiel.

d. **Comments.** SEAC plays the damage that is caused by enemy air attacks. It does not generate missions. The missions (size, number of planes, time, etc.) are all developed offline, based on information found in applicable threat documents.

12. **SOF Threat Data File.** The SOF Threat File defines a possible scenario of enemy special purpose or operations teams and units directed against COMMZ targets.

a. The file is based on information found in an intelligence report.

b. **Content and format.** Figure C-9 displays the content and format of the SEAC SOF Threat File.

c. **Classification.** The SOF Threat File is classified according to source classifications.

DATA FORMAT FOR THE AIR THREAT FILE

Data Item	Start Column	Length	Type	Comments
Day	1	2	Integer	--
Hour	4	2	Integer	--
Target	7	4	Text	Target's GEOLOC
Origin	12	4	Text	Threat airbase
Fighter/Bombers	17	4	Integer	Number of planes
Bombers	22	4	Integer	Number of planes
Readiness	31	5	Real	Operational readiness
Target preference	?	*	Real	Targeting information optional

?: = free format.
*: = no fixed length.

Figure C-8

DATA FORMAT FOR THE SOF THREAT

Data Item	Start Column	Length	Type	Comments
Time	1	5	Real	--
Team type	7	5	Text	Name of SOF team
Size	13	5	Integer	Number of teams
Target (Installation)	21	4	Text	GEOLOC of target
Target (Mapcell)	27	4	Integer	Mapcell

Figure C-9

d. **Comments.** The SOF file is much like the Air Threat File. SEAC only evaluates the proposed scenario in terms of the damage to facilities the threat's activities are likely to produce.

13. **Orders File.** The data most important to analyzing engineer support is found in the Orders File. This file enables the analyst to insert engineer projects, manipulate engineer units, and change the list of mapcells that comprise the COMMZ. The bulk of records in the file concern engineer units. SEAC does not use the TPFDD directly for engineer unit assignments. SEAC's hierarchical representation of the COMMZ (simulating engineer command and area support relationships) and explicit unit representation (each unit is individually monitored and controlled) permit the user to assign and move engineer units in a manner that closely parallels how they would actually be used. While most facility requirements result from checking installation and unit associated planning factors, some facility needs can only be identified through independent analysis. The Orders File is the means within SEAC to enter these projects. And finally, scenarios theorize movement of the forward edge of the battle area (FEBA). As this occurs, the COMMZ may expand or contract as the case might be. Each mapcell defined for the theater has a flag indicating whether it is to be considered in the COMMZ. This flag can be controlled by ORDER directives.

a. **Source.** Since there are three types of records to be constructed, there are multiple sources for this file.

(1) Special engineer projects or facility requirements based on OPLAN directives, sponsor input, or analysis of logistic support structure requirements are inserted into the Orders File at the time and for the place they are needed.

(2) Engineer unit directives are largely derived from the TPFDD (or its derivative, CESPG's TROOP File), and from the OPLAN under study, especially the engineer annex. These directives are further supplemented by sponsor guidance and study team analysis.

(3) The final type of record is used to indicate which mapcells are to be considered within the area of responsibility of the COMMZ. If the mapcell is flagged as being in the COMMZ engineer area of responsibility, then workload is determined and available capability is applied. If flagged otherwise, construction and damage requirements are calculated, but tasks are

not generated and work is not performed. The mapcell order record is how the flag is changed. (NB: The default mapcell flag assumes that is to be included in all calculations.)

b. **Content and format.** Figure C-10 displays the various formats of order file records.

DATA FORMAT FOR THE ORDERS FILE

Data Item	Start Column	Length	Type	Comments
Time:	1	5	Real	--
Record type	10	1	Text	Unit, project, area
Nation	12	1	Text	--
Service	14	1	Text	--
Unit:				--
UTC	16	5	Text	--
Strength	21	4	Integer	--
From location	26	4	Text/Integer	GEOLOC, Region, empty
To location	31	4	Text/Integer	GEOLOC, Region, empty
Project:				
Facility	16	4	Text	CATCODE
Quantity	22	8	Real	--
To location	31	4	Text/Integer	GEOLOC, Mapcell
Unit of Measure	36	2	Text	--
Mapcell:				
Mapcell	16	5	Integer	--
Include	26	1	Boolean	--

Figure C-10

c. **Classification.** This file is classified since engineer unit locations and timing are largely derived from classified sources -- OPLANs and TPFDD.

14. Priority/Policy File. One of SEAC's key features, to "assess" the adequacy of engineer planning and support, is the priority system. Basically, this is a categorization scheme that subjectively appraises the effect of non-performance of engineer tasks. SEAC combines their priorities with policies, which control what and how much of engineer work is done.

a. **Source.** The sponsor is the ultimate source of engineer task priorities. ESC, as the originator of the concept, does take an active role in describing the objective of the process and reviewing the sponsor's elections for consistency and reasonableness. Policy assignments get into more tenuous territory -- it is with less assurance that one can say the HN will repair 30 percent of war damage, before the magnitude of that damage is known. Policies are more likely to be based on either assumed levels of support, or support agreements.

b. **Content and format.** The format of the Priority/policy file is found in **Figure C-11**. It is an ordered file in that the records following the keyword records are assumed to apply to the work category the keyword denotes.

DATA FORMAT FOR PRIORITY/POLICY FILE

Data Item	Start Column	Length	Type	Comments
Task Group	1	5	Text	Indicates construction, damage, etc.
Facility class	11	4	Text	Four-character facility subCLASS id
Installation class	11	4	Text	Four-character install. subCLASS id
Country	21	1	Text	
Priority	26	1	Text	v = vital, c = critical, etc.
Policy	32	5	Real	Percent of work to be done by military resources.

Figure C-11

(In the first version of the model, MAINT (maintenance) and RESTR (restoration) are acceptable entries in the file, but there is no operative logic or code installed in SEAC for those functions.) Another feature is how SEAC interprets missing nation-task-class entries. It is not necessary to enter all installation CLASSES for construction, or facility CLASSES for damage repair. The model assumes, however, that a non-entry means no work is to be generated for that country-task combination. This might apply if, for

example, a decision was made not to repair revetments or that the host country has sole responsibility for port construction.

c. **Classification.** This file is considered UNCLASSIFIED.

d. **Comments.** The priority system that this file used in SEAC is becoming a common and preferred means to rank engineer work prior to allocating resources.

LAST PAGE OF ANNEX C

ANNEX D
USERS GUIDE TO SEAC

ANNEX D
USERS GUIDE TO SEAC

<u>Paragraph</u>		<u>Page</u>
1	Purpose	D-1
2	Scope	D-1
3	Using SEAC	D-1
4	Preparing the Database	D-3
5	Modifying SEAC	D-13

Figure

D-1	SEAC Analysis Process	D-3
D-2	SEAC INPUT/OUTPUT STRUCTURE	D-4

1. Purpose. This annex provides guidance to analysts who plan to use SEAC to conduct analyses of engineer work in the communications zone (COMMZ).

2. Scope. The philosophy behind SEAC is first discussed. Then the focus settles on the purpose and preparation of data used in the model.

3. Using SEAC. SEAC was developed as an in-house tool of the Engineer Studies Center (ESC). ESC's experience and mission are to analyze important problems connected with policy development and decisionmaking. Because of its comparatively small staff, ESC has always been reluctant to assume proponenty for a particular model whose operation and maintenance would consume scarce resources. ESC has on occasion, nonetheless, developed computer-based systems when necessary to successfully accomplish studies. Where possible those systems are adapted to use on other similar studies. The same multifunctional preference is evident in the design philosophy of SEAC. Although SEAC was created to accomplish a particular study, it was designed as a tool that would be adaptable to a variety of problems involving COMMZ engineering. Rather than designing it to be a "black box," ESC approached it as the archetype for COMMZ assessments, both general and specific. SEAC can be viewed as an analytic workbench where the problem at hand will indicate what if any changes are necessary.

a. **Models as Black Boxes.** Most models, systems, and games (MSGs) are designed as stand alone systems that are the proverbial black, or closed box to most users. A user assembles the required input data, executes the model, and examines the results. How the model determines those results is very often of little concern to the user; it usually suffices that he only knows the general theory behind the model. The black box approach has justifications such as: it avoids error and comparability questions that arise when internal programs are changed from one problem to another; it usually assigns one group responsibility for program maintenance, improvements and corrections; and finally, analysts are able to use (i.e., prepare the input and analyze output) the model much faster if they are not compelled to understand the internal manipulation of each datum.

b. **The SEAC Process.** SEAC was developed as a tool to analyze engineer work in the COMMZ. It simulates the generation of engineer task requirements and the assignment of programmed engineer resources to meet that workload. SEAC followed the traditional steps of model development: analysis of the system, implementation of the model, verification, and validation. Unlike many model development efforts, however, it did not arise in a vacuum. Concurrent and intertwined with SEAC's development was the study¹ that spawned it. As an independent study, it too had identifiable phases: problem definition, data collection, analysis, and prescription. One might therefore think of SEAC being developed and, in the future, operating in a two-dimensional space: one dimension or axis is study accomplishment; the other is model modification and verification. Figure 1 portrays these steps as well as the feedback that occurs. Since ESC anticipates that SEAC will evolve to meet new study applications, this two-dimensional environment is expected to continue to characterize SEAC. New studies will introduce new objectives and processes that will require a new cycle of analysis and model validation. ESC, therefore, considers SEAC to be an "open" box model (although of course a user could use the model without changing or adding unneeded details).

¹Engineer Assessment, Korea: Communications Zone Analysis (USAESC, August 1987).

```

graph TD
    PD[Problem Definition] --> DC[Data Collection]
    DC --> SS[System Synthesis]
    SS --> MV[Model Verification]
    MV --> MV
    MV --> VV[Model Validation]
    VV --> VV
    VV --> A[Analysis]
    A --> PD
    A --> DC
    A --> MV
    A --> VV
    A --> C[Conclusions]
    C --> C
    C --> MG[Modeling Goals]
    MG --> SA[System Analysis]
    SA --> SS
    SA --> VV
    SA --> C
    SA --> MG
  
```

The flowchart illustrates the modeling process. It begins with 'Problem Definition', which leads to 'Data Collection'. 'Data Collection' leads to 'System Synthesis', which leads to 'Model Verification'. 'Model Verification' has a feedback loop to itself and leads to 'Model Validation'. 'Model Validation' has a feedback loop to itself and leads to 'Analysis'. 'Analysis' has feedback loops to 'Problem Definition', 'Data Collection', 'Model Verification', and 'Model Validation'. 'Analysis' also leads to 'Conclusions'. 'Conclusions' has a feedback loop to itself and leads to 'Modeling Goals'. 'Modeling Goals' leads to 'System Analysis', which leads to 'System Synthesis' and 'Model Validation'. 'System Analysis' also has feedback loops to 'Modeling Goals' and 'Conclusions'.

4. Preparing the Database. SEAC has a rather prodigious appetite for data. Figure D-2 shows the model in terms of input and output data. The upper-left portion identifies the files and data that the model uses. Some data files are culled, or adapted from standard sources and require only minor changes, if any, on the part of the user. Other files are unique to SEAC and, therefore, must be created by the user. If the analyst is examining a theater that has been the subject of a previous study, some of the latter files may possibly be recycled with only minor changes or fixes. As with most computer systems, the data structure is probably best learned by examining existing files and adapting them to the problem at hand. The guidance that follows attempts to give information, as does Annex C, which would enable an analyst to assemble any of the SEAC files. (A user is also urged to scan the CLASS parameter and attribute definitions found in Annex B to understand the data structures used in the model.)

SEAC INPUT / OUTPUT STRUCTURE

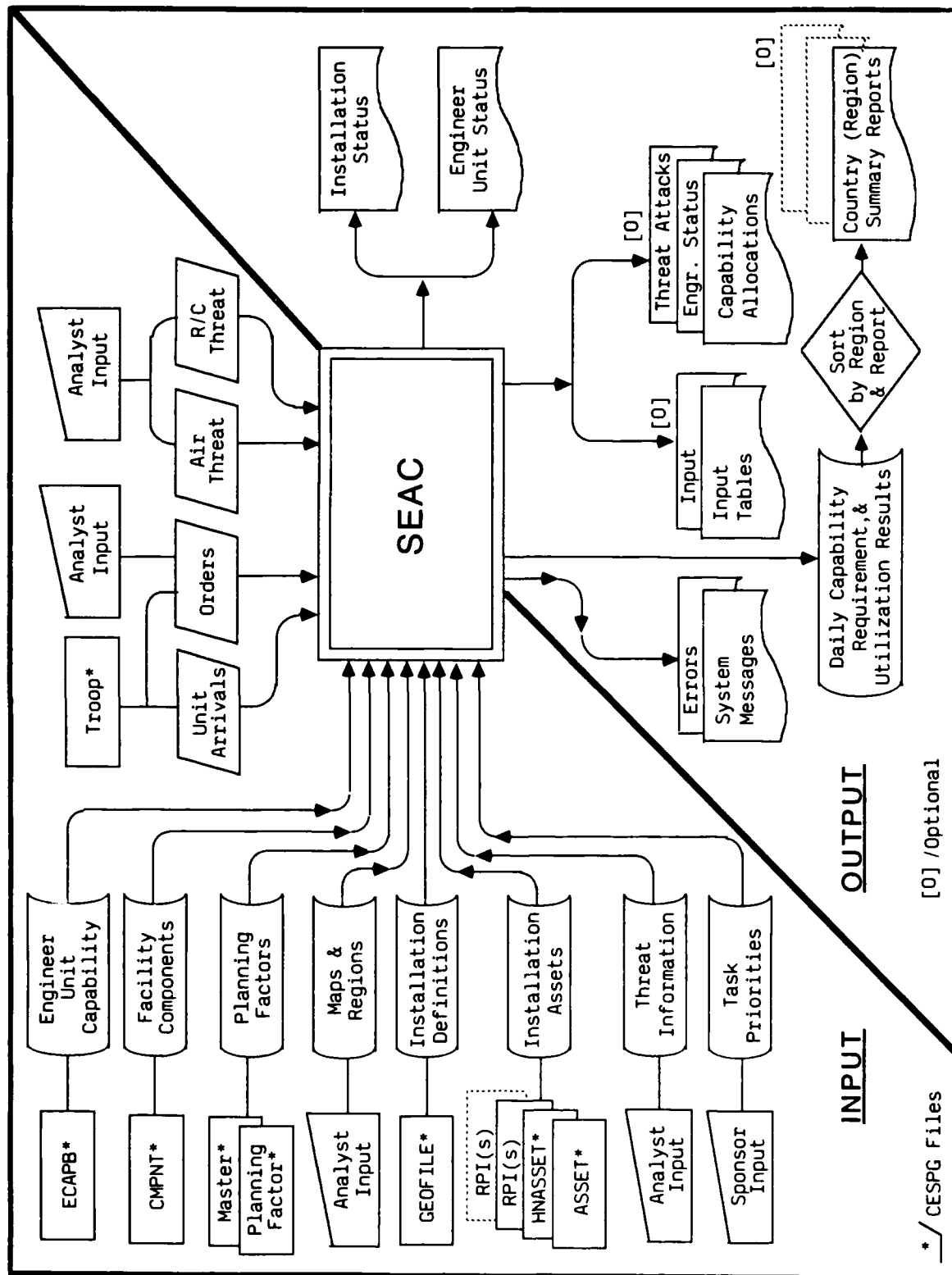


Figure D-2

a. **Designing the Theater.** Representing the theater in a way that accommodates both geographic and scenario requirements was one of the principal objectives of SEAC. Other engineer models are frequently faulted for their rather restrictive treatment of operations. The Civil Engineering Support Plan Generator (CESPG), for example, groups geographic locations together to form something called a base complex. This base complex is a notionalized super installation which contains all the facilities located within the area it encompasses and all the engineer units destined for those locations. Each base complex is treated in isolation, giving rise to situations where a complex with unsatisfied requirements might be next to one with excess capabilities. Nor can units typically be moved between complexes. In contrast to this, SEAC provides a more realistic treatment of both individual installations and engineer units. It does this through a representational hierarchy of regions, mapcells, and installation objects.

(1) **Regions.** Representing the area support mission of Army engineers was the stimulus for creating the region. Engineer units may directly support a specific project, unit or installation, or they may be in general support, thereby allowing task organizations to support a wider geographic area containing many installations or units. Defining the operational or support locale for a unit when it might, at one time, have been a single facility, or another time an entire province, or possibly a theater Army area, posed a dilemma. The region hierarchy scheme was developed to solve this quandary. A region is explicitly defined by the number of regions (subregions) that belong to it, and the region to which it belongs. Note that a region may have many subregions but only one superior region. Regions (subregions) may also contain mapcells (infra). The important attribute of regions (subregions) is that engineer units can be assigned to them. When thus assigned, an engineer unit can support any engineer requirement that arises in the region, its subregions, and the mapcells they might contain. Definition of the regional hierarchy should therefore be based on anticipated areas of responsibility and the anticipated operational use of engineers in the theater. If the plan calls for a unit to provide support to several Army installations and an Air Force base, then there should be a region that contains (directly or derivatively) those installations.

(2) Mapcells. Regions do not provide a complete description of the theater. Mapcells complete the theater hierarchy with a unique partitioning of the COMMZ that links installations with regions. It uses an abstract, user defined representation that divides the theater into cells to which installations and MSR facilities are assigned. Mapcells do not preserve any relative location data; a mapcell has no knowledge of its neighbors. The region is relied upon to provide the locality information. Defining the mapcells in a theater is a subjective process, but there are guidelines. Typical considerations that influence mapcell definitions are: installation groupings, locations of major MSRs, important political or organizational boundaries (which regions will treat) and, of course, geographical features. The COMMZ may be thought of as a network of nodes (installations) and links (MSRs). Mapcells can be used to segregate those areas from unimportant geographic areas. Needless to say, the starting point is a map of the theater that contains terrain and installation data. Developing the mapcells then becomes a matter of balancing the tangibles (geography, installations and facilities) with the intangibles (plans and units), with the emphasis on the former. There are many possible partitions, any one of which can be suitable. One alternative is to simply overlay a symmetric grid onto the theater and make each square or hexagon a mapcell. SEAC does not, however, demand that the user overlay a grid. Unlike regions, mapcells do not have engineers associated with them; instead, they have installation and MSR data. By defining regions according to which mapcells they contain, SEAC is able to set up different regional trees for each nation being played in the analysis.

(3) Installations (GEOLOCs). Installations are the camps, airbases, communications sites, etc., that are found throughout the COMMZ and are militarily significant. They are identified by a geographic location code (GEOLOC). The user can indicate the type of installation for which SEAC will apply the appropriate factors and treatment. Currently, there are eight classes of installations: airbases, camps, ports, ADA sites, electronic sites, hospitals, depots, and others. The user should refer to the CLASS descriptions for these installations found in Annex B. Identifying the appropriate installation class may require sleuthing through various lists of installations (e.g., the Army location file). There may also be entries for the installations in the factor file. Where several installations appear to

be collocated, they can be grouped together to permit their assets to be pooled.

(4) **Facilities.** Engineer activities in the COMMZ are basically to support the logistic structure in the theater rear. Engineers may be called upon to build, repair, or maintain roads, runways, troop housing, maintenance shops, etc. Within SEAC, facilities are classified into various groups, primarily to provide common or specialized handling of particular tasks. A good example is that war damage repair on a building is based on the percent of the building that is damaged, while damage to an airbase runway is measured by the number of craters that exist (and provision is also made to classify craters into different groups to simulate the number necessary to attain a minimum operational strip). There are five general classes (for surface, petroleum, buildings, utilities, and transportation components). These classes, in turn, contain varying numbers of subclasses to permit the inclusion of other special data and procedural logic. (See also paragraph 4.g.)

(5) **COMMZ Hierarchy.** The region/mapcell/installation hierarchy described above is the environment in which requirements (facility needs) and capability (engineer units) arise and interact. One can visualize the system as having several planes: in the middle is the mapcell plane in which the theater is uniquely partitioned; below it is the installation plane on which all the installations are found, each one linked to a particular mapcell above; below that level are the facilities which belong to the installations; and above the mapcell plane is(are) the region plane(s) which group(s) adjacent mapcells (or other regions) into areas based on support concepts. An additional feature of SEAC is its ability to incorporate multinational participation. Within the COMMZ hierarchy, this is done by having parallel regionalizations for each country being played. The one thing they have in common is that they all point to the same, unique mapcell plane. The model keeps track of and distinguishes installations of different countries at the mapcell level.

b. **Threat Assessment.** Another design goal of SEAC was to generate damage to facilities closer to the actual capability of the enemy threat. The CESPG levies damage, but only until D+30. It is also rather clumsily done by entering daily percentage damage figures of particular CATCODES within a

base complex. ESC's methodology is much more detailed and is an intrinsic part of SEAC. The threat is defined in terms of what delivers damage -- either planes or infiltration units. These, in turn, are defined by what they can carry that can damage a facility. When a bomb is dropped, or demolition materiel used, the amount of damage that it produces against the facility is calculated. It must be emphasized, however, that SEAC is not a wargame -- missions against the COMMZ are scripted offline to the model.

(1) Instruments. Damage generation is a direct result of the enemy's ability to deliver ordnance against a target. Planes are defined by a likely ordnance configuration. Intelligence sources are the essential source for this information. Infiltration units, used in rear area attacks, are similarly defined. For SEAC's purpose, the only germane threat data concern the amount of weapons and demolition materiel that these infiltrators are likely to carry, and which can be used to destroy or damage facilities.

(2) Air. The current implementation of the air-attack model uses preplanned sorties. This was acceptable for the operational plan (OPLAN) used in SEAC's initial application, but may present problems under different scenarios where the air war may be a different length. The logic of the air war is described in the main paper of this document, and the format of the air mission file is in the data Annex C.

(3) Ground. Like the air war, the ground (infiltration) threat also follows a preplanned script. It is somewhat different, however, because there is more uncertainty surrounding the infiltrating units. They may, or may not reach their target; if they miss their target initially, they could still attack it at a later time; they could also be destroyed enroute before reaching a target. More important is the fundamental difference between air and ground instruments: planes can and are rescheduled for attacks, while ground units are considered expendable -- once they use their ordnance, they are no longer considered by SEAC. Like the air module, the ground module is only concerned with engineer repairable damage. It does not evaluate the disruption that infiltrators would likely cause. The ground submodel is described in the main paper, and the data formats are found in data Annex C.

c. Orchestrating the Scenario. Until SEAC was developed, there was no engineer COMMZ model that could directly manipulate engineer resources in response to scenario events. CESPG takes its engineer resources as it finds

them on the Time Phased Force Deployment List (TPFDL). (Special engineer situations in CESPG appear to be handled by making "kludgey" changes to the data base, since the model logic is not easily accessed.) The CESPG accepts the destination GEOLOC as the permanent duty station, and the unit's resources are added to those already available to the base complex that contains the GEOLOC. SEAC, on the other hand, preserves the identity of engineering units to allow better control of movement and capability within the COMMZ. Changes in unit status are made through directives entered in the ORDERS file. In addition to manipulating engineer units, ORDERS also is used to insert projects into the COMMZ and to redefine the physical limits of the COMMZ. The latter would be used if the theater expanded or contracted as a result of changes in the forward edge of the battle area (FEBA). The combination of identifying individual engineer units within the model, being able to change their location (area of responsibility), and the regionalization of the theater described above are among the major improvements in how SEAC manages engineering capability.

(1) Engineer unit taskings. The starting point for constructing unit taskings is to identify US engineer units found on the TROOP file (or TPFDD from which it is derived). Ideally, all engineer unit records should be extracted and examined. The unit designator, service affiliation, destination, arrival date, and COMMZ identifier (which distinguishes among those Army units who will probably deploy forward, and those that stay in the COMMZ) are the items that must be analyzed. If there are engineer units that do not appear on the engineer capability file (ECAPB), the user must either drop the unit, find an equivalent unit ECAPB entry, or establish a new ECAPB entry. Before making this check, the ECAPB file should itself be checked to assure it reflects current tables of organization and equipment (TOE) and tables of distribution and allowances (TDA). Unit assignments may be either to installations (identified by GEOLOC) or to regions. The current convention is to assign non-US Army units to the installation designated by their TPFDL destination. Army units, given their general support mission, are usually assigned to a region that contains their destination. There are exceptions. An Army utility team would go to the installation, and Army port construction companies would go to port installations. Remember that engineer support is top-down -- a unit assigned to an installation will only support that

AD-A196 529

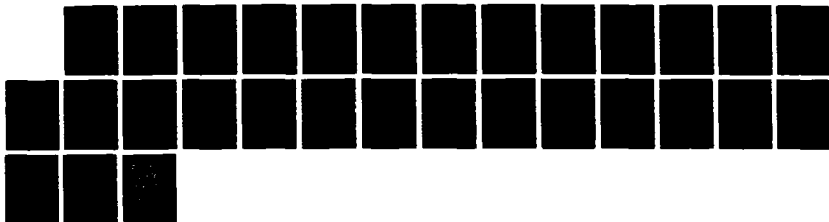
SIMULATED ENGINEER ASSESSMENT OF THE COMMUNICATIONS
ZONE MODEL (SEAC) (DO. (U) ARMY ENGINEER STUDIES CENTER
FORT BELVOIR VA R MALAYKO JUN 88 USAESC-R-88-3

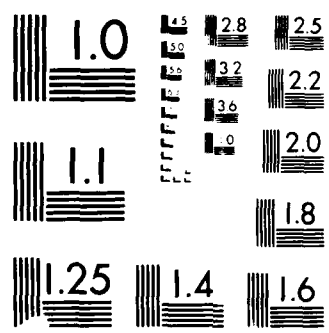
4/4

UNCLASSIFIED

F/G 25/5

NL





UTION TEST CHART

installation, while a unit assigned to a region will support all requirements within that region, the mapcell it encompasses, and the installations they, in turn, contain. Specific host-nation contractors (who are coded as US units) can be inserted when the user knows of specific host-nation capability and project responsibility. The user will also have to identify engineer units destined for the FCZ which normally would be excluded from the list of available COMMZ units. In addition to assigning units, SEAC can move and extract as well. The user can, therefore, simulate unit movements that are either intra- or inter-theater. For example, if execution of the model shows large capability and requirement mismatches, the user could insert movement orders for underutilized units. (The model does not reallocate units automatically.)

(2) Special projects. Workload factors do not generate all required facilities. There are some facilities, and possibly even whole installations, that are required but cannot be generated from simply processing the TROOP list. These projects are coded by location, category code, size, and occurrence and placed in the ORDERS file. MSR projects designated for mapcells are scheduled, since they presumably arose from separate analysis. Facilities indicated for installations increase appropriate requirements but are still subject to whether there are enough assets on-hand before a task is scheduled.

(3) FEBA movement. Most wars are not stalemates. Territory expands or contracts depending upon the movement of the FEBA. SEAC is not a combat model and, therefore, is not cognizant of changes to the location of the COMMZ. To account for any such changes, however, SEAC employs an ORDERS file record to indicate whether a mapcell is in a COMMZ area of responsibility or not.

d. Unit Arrivals. Most requirements result from the arrival of deploying troops in the COMMZ. SEAC relied on CESPG's TROOP file for that information. That file was used because it theoretically is an extract of the TPFDD that has been carefully reviewed and corrected. The TPFDD can be used directly but that also means the user must eliminate equipment records, purge bad records, and remove units not involved with the COMMZ (primarily for run time efficiency). For a force list of thousands of units, this can take quite a bit of time, given the TPFDD's size and sometimes sizeable error rate.

e. **Engineer Unit Capability.** Engineer units in SEAC provide the manpower resources to perform tasks. The source of information on what a unit's capability is comes from the engineer capability file used by the CESPG. This file should be reviewed, however, since units change and differences exist between TDAs and TOEs. This file will usually be augmented by information about indigenous contractors and other national military engineer units.

f. **Planning factors.** Facility requirements are primarily generated by factors. Units, people, equipment, and installations can all demand facilities. Most people and equipment factors come from current Joint Chiefs of Staff (JCS) guidance.² Unit related factors can be found on the CESPG MASTER file. Installation factors are a unique feature of SEAC. The CESPG has factors for airbases, but does not identify other installation types to permit special treatment (the base complex limitation). Installation factors allow the user to define minimum essential facilities for each class of installations. (Another benefit is the default treatment of installations at day 0. The current version of SEAC assumes that required facilities, predicated on the factors, exist at installations at the beginning of a scenario. This was done, in particular, to create facility inventories at non-US installations for which no asset information could be obtained.) Since factors change with theater and scenario, it is incumbent upon the user to review the factors and assure their appropriateness.

g. **Facility Components.** Requirements are expressed in terms of amount of facilities needed. That might be kilowatts for generator needs, square footage for operational facilities, or gallons for water production. Requirements are translated into engineer work by finding a facility component which best satisfies these needs. The component data provides the skill effort needed to construct or repair a given sized facility, expressed in terms of daily hours of needed horizontal, vertical, and other work hours. The information comes from the facility component systems of each US service.³ This is the same data (found on the CATCODE file) that is used by the CESPG.

²*Planning Factors For Military Construction in Contingency Operations*, Enclosure to Joint Chiefs of Staff Memorandum, MJCS-201-81 (13 October 1981).

³*Army Facilities Components System User Guide*, Technical Manual 5-304 (Headquarters, Department of the Army, October 1979).

The only difference is that ESC does not include multiple components within a category code/project class. In the initial version of SEAC, three components were selected for temporary construction, initial construction, and damage repair. The latter is construed to mean repair of temporary standard facilities. ESC culled most of these data from the CESPAG component file. In some cases changes were made to suspect entries. Some features of SEAC required making pseudo components (for MSR tunnel and bridge entries). The components chosen influence the workload to different degrees. Such things as satisfying electric power requirements with a few large, rather than many small generators can substantially change engineer requirements. The user should inspect the CATCODE file to assure that component data is correct for the theater location (i.e., frigid, temperate, or desert) and that the facilities correspond to the concept of operations.

h. **Pre-existing assets.** For those theaters where US units are forward deployed (Europe and Northeast Asia), the sizeable inventory of existing US installations and facility assets must be accounted for before new requirements can be estimated. The service maintained, real property inventories (RPI) should make this an easy task, but experience has shown those files to be rather turbulent. A user should closely examine those RPI files or the CESPAG asset files to ensure the data is representable. (NB: Recycling facilities such as schools into administrative buildings or billets is not done automatically by SEAC. ESC developed a program that converts assets from peacetime to possible wartime use by translating Army Facility Component System (AFCS) codes into JCS CATCODE equivalents.)

(1) **Task prioritization.** Engineer work is not all equally important. Some facilities are more important than others, and to assure that effort is applied to these facilities, SEAC prioritizes projects. The relative importance (see the discussion of task prioritization in the main paper) of work for each country in the analysis can be included. Construction priorities are presently derived from the type of installation at which the facility is being built; damage repair priorities are based on the class of facility being repaired. Restoration and maintenance, if added work, would similarly be based on installation and facility classes. It is generally the sponsor of the study that sets the priorities. (NB: The priorities file can also be used to not only rank, but also preempt certain tasks. First, if a

type of work, e.g., repairing railroads, is not in a country's priority list, the implication is that work is not done. Second, the file also contains an estimate on the amount of effort that the host nation is expected to provide. Thus, if there is an entry for repairing railroads, but the percent responsibility was set at 10 percent, the result is that the host nation would do 90% of the work. (Needless to say, tying construction priority to an installation means that a wide range of facilities all receive the same priority. ESC anticipates that one of the first changes to SEAC will be to link construction policy to facility class as is done for war damage repair.)

5. Modifying SEAC. SEAC provides a software framework to analyze engineer workload. The earlier description of SEAC's philosophy emphasized the intended mutability of the model. Rather than contorting a problem into a form that SEAC requires, the programmer/analyst can (is expected to) change or extend the model. This is, to a large degree, a result of the environment in which SEAC was created. The features of the implementation language, SIMULA, were, in fact, specifically intended to support the development of problem-oriented concepts. To be able to make these changes or additions, however, requires the user to understand SIMULA first and SEAC second. There is little guidance that can be offered to someone considering such changes since there is no shortcut to understanding the software.

ANNEX E

SEAC REPORTS AND MESSAGES

ANNEX E

SEAC REPORTS AND MESSAGES

<u>Paragraph</u>		<u>Page</u>
1	Purpose	E-1
2	Scope	E-1
3	SEAC Reports	E-2

Figure

E-1	CATCODE/Component Error Messages	E-3
E-2	Reception, Staging, and Deployment Errors	E-4
E-3	ORDERS File Error Messages	E-4
E-4	Threat Team and Plane Weapon/Ordnance Messages	E-5
E-5	Air War Messages	E-6
E-6	Threat Mission Summary Report	E-6
E-7	SEAC Daily Cycle Change Messages	E-8
E-8	Errors Identified During GEOLOC File Processing	E-8
E-9	Contents of the SEAC Monitor File	E-9
E-10	Engineer Capability Data	E-10
E-11	SEAC Planning Factors	E-11
E-12	Category Code/Component Listing	E-12
E-13	Unit of Measure Data Listing	E-13
E-14	Installation GEOLOC and Type Assignments	E-14
E-15	Mapcell-Installation Cross-Reference	E-15
E-16	Report of Installation/Facility War Damage Events	E-16
E-17	Engineer Resource Tasking Results	E-18
E-18	Installation Status Report	E-19
E-19	Theaterwide Engineer Unit Status Report	E-20
E-20	Capability Report for Regional Summary	E-21
E-21	Work Submission Report for Regional Summary	E-22
E-22	Requirements Report for Regional Summary	E-22
E-23	Utilization Report for Regional Summary	E-23

1. Purpose. This annex presents examples of the various messages, runtime information, and reports currently provided by SEAC.

2. Scope. The SEAC model is relatively self-contained, i.e., it does not rely on other programs for pre- and post-processing tasks. The user prepares the data files according to required formats, and then lets the model take over. It reads the input files, checks syntax, flags questionable or erroneous data, constructs efficient look-up tables, reports on key actions

and milestones during the simulation of activities and, upon completion, constructs report files summarizing the results. This annex will touch upon the output that accompanies each of these phases.

3. SEAC Reports. One of SEAC's design objectives was to simulate engineer requirements and capabilities in as much detail as data and computational limits permitted. SEAC takes maximum advantage of the capabilities afforded by the modeling language's dynamic memory allocation and the operating system's virtual memory management. Consequently, there is a considerable amount of information, both interim and permanent. To save the state of the model every simulated day, or worse yet, at every major change in status, would quickly overwhelm core, virtual, and auxiliary memory. To wisely use computer time and memory, became as important a design need as the functional modeling. SEAC adopted a rather flexible approach to reporting needs. Output falls into several categories: automatic (such as error messages and certain summary reports at normal termination), optional (printing input data records or showing details of internal processes), and definable (indicating the level or area for which summary reports are needed).

a. Messages. SEAC messages cover a variety of purposes: errors, warnings, interim results, and status. Most messages, though terse, give a sufficient indication of the condition or fault that has been encountered and are listed below but are not individually discussed.

(1) CATCODE processing. CATCODEs are the means by which facilities are identified within SEAC. Managing facility data occupies the bulk of SEAC's time. Assets, planning factors, component data, damage, tasks, and projects all make use of CATCODE identifications to manage facilities. Needless to say, the opportunities to introduce erroneous data are manifest. The model checks CATCODEs whenever they enter SEAC's environment. Bad CATCODEs and CATCODEs with invalid or inconsistent units of measure are rejected and reported. (Valid CATCODEs, incorrectly used, are the responsibility of the user.) The messages shown in **Figure E-1** are examples of such errors, and also indicate that SEAC tries to identify the internal procedures or processes where the error surfaced: "NEEDS" indicates a mismatch that occurred during planning factor application; "ASSETS" indicates that the error occurred while processing the facility asset file during the setup phase of the simulation.

CATCODE/COMPONENT ERROR MESSAGES

```
....PROP/NEEDS..UOM MISMATCH..841C-KG
....PROP/NEEDS..UOM MISMATCH..851A-MI
....PROP/ASSETS..UOM MISMATCH..841B-GA
....PROP/ASSETS..UOM MISMATCH..214A-VE
....PROP/XREF..MISSING COMPO..832A
```

Figure E-1

(2) Reception, staging, and deployment (RS&D). The identification and processing of individual units enables SEAC to determine facility requirements at the installation level. Much of the routing and movement of units is handled by logic that controls the RS&D process. The code makes every attempt to receive and move units in a realistic manner. Nonetheless, errors and undefined data are encountered which will prevent some units from being included. Rather than terminating execution, the condition and the data record that caused it are entered into the log file as shown in Figure E-2 for post execution review by the user.

(3) ORDER directives. The ORDERS File contains a variety of record types: engineer unit changes, construction or facility requirements, and COMMZ changes. The variety introduces an opportunity to make many different types of errors, all of which must be checked for by SEAC. Figure E-3 shows the various errors that are detected.

(4) Threat definition. War damage is a direct consequence of the demolition and ordnance materiel that is carried by infiltration teams and enemy aircraft. Platforms can be configured to carry different ordnance loads. To verify that the appropriate data was read and accepted for the threat, SEAC creates the messages, shown in Figure E-4, that indicate what and how much of each definition was successfully processed.

(5) Threat air attacks. Air attacks are grouped by strikes and missions within that strike. When a strike begins, the status of enemy planes is first established. This determines how many planes are available, how many of those are not operationally ready, and how many are currently undergoing repair. Planes are then assigned to each mission. If there are not enough planes in the available pool to meet mission quotas, a message indicates the

RECEPTION, STAGING, AND DEPLOYMENT ERRORS

```
..RSD.. IN PLACE UNIT DEST MISSING FOR ENTRY --
AAAAA UNIT2ZPXE 222 -99 A
..RSD.. POD NOT DEFINED AS APOD/SPOD FOR ENTRY --
AAAAA HELOBALVP 2TKDX 2 A
..RSD.. NO SERVICE RECEPTION CTR FOR ENTRY --
AAAAA HELOBALVP 2TKDX 2 A
..RSD.. POD NOT DEFINED AS APOD/SPOD FOR ENTRY --
APUSH PUSHYEVAT 100TKDX 2 A
..RSD.. NO SERVICE RECEPTION CTR FOR ENTRY --
APUSH PUSHYEVAT 100TKDX 2 A
```

Figure E-2

ORDERS FILE ERROR MESSAGES

```
..CHANGE ERROR..(FROM REGION NOT FOUND)--
-1 U A A 4X7ME 99
..CHANGE ERROR..(UNIT NOT IN FROM-LOC)--
-1 U A A 4X7ME 99
..CHANGE ERROR..(UNIT NOT IN FROM-LOC)--
00000 U A A 4X7ME 1 19
..CHANGE ERROR..(UNIT NOT IN FROM-LOC)--
00000 U A A 4WXCG 1 99
..CHANGE ERROR..(UNIT NOT IN FROM-LOC)--
00000 U A A 4WXCG 1 XXXX
..CHANGE ERROR..(TO INSTAL NOT FOUND/MATCHED)--
00000 U A A 4X6AA TVRL SMYU
..CHANGE ERROR..(^...PUT ARMY ENGR INTO REGION)--
00000 U A A 4X6AA TVRL SMYU
..CHANGE ERROR..(UNIT NOT IN FROM-LOC)--
00000 U A A 4WXCG 1 EVAT
..CHANGE ERROR..( MAPCELL INDEX)--
2 P A A 125A 10000 63 LF CONT PIPE 1A
```

Figure E-3

THREAT TEAM AND PLANE WEAPON/ORDNANCE MESSAGES

T	AN22		PLSTC 60.0	^..SPF..MUNITION NOT FOUND--TM CENSORED				
A	IL28	B	800 450	FB250	4FB250	2FB250	2	^ MUNITION NOT

Figure E-4

shortfall. A mission defines how many bombers and fighter bombers are to proceed from the threat airbase to an installation target. An option also allows the user to designate facility class targeting if different from the default targeting scheme. Finally, upon completion of each mission, the number of planes that reached the target, the number of planes destroyed, and the number of planes damaged are given as shown in Figure E-5.

(6) Threat summaries. When the air and ranger/commando threat files have been exhausted, and all missions completed, SEAC summarizes the overall conduct of threat actions with short tables like those shown in Figure E-6. For the air threat, it gives the total number of missions processed, and a more detailed breakout, for fighter/bombers and bombers, of how many planes were scheduled to go on the missions, how many actually went, how many were killed or damaged before or after reaching their targets, and how many actually completed their mission objective. The even more cryptic summary of commando operations which is interpreted to mean that: there were 27 planned missions; eight teams were destroyed enroute; three attacked their target installations; no teams were sent initially against area targets (i.e. sent to attack any MSR or Installation facilities that it might encounter); 16 teams initially missed their installation targets (and by convention are then placed in the mapcell area of the target); three of those teams were later destroyed before attacking any target; and the remaining 13 teams attacked installations in their area or adjacent areas to which they moved.

AIR WAR MESSAGES

NEW STRIKE BEGUN 06

POOL	BASE	BOMBERS	FIGHTERS	REPAIR	(T= 0.25)
	NK01	35/ 5	188/ 12	0	

BOMBER SHORTAGE FOR TARGET TVRK

MISSION REPORT	1	NK01 TO SMYU	5	0	0
MISSION REPORT	2	NK01 TO ESXM	12	2	1
MISSION REPORT	3	NK01 TO VHPY	13	2	0
MISSION REPORT	4	NK01 TO WNHQ	14	0	1
MISSION REPORT	5	NK01 TO TVRK	5	0	0

Figure E-5

THREAT MISSION SUMMARY REPORT

+++ AIR WAR MISSIONS = 11

SCHED	ACTUAL	KILLED		DAMAGED		ATK
		PRE	POST	PRE	POST	
85	75	2	5	1	6	64
66	66	3	3	3	2	57

RANGER COMMANDO SUMMARY:	<u>PLAND</u>	<u>X-IN</u>	<u>IIATK</u>	<u>AASG</u>	<u>ADFLT</u>	<u>X-GND</u>	<u>MSR</u>	<u>I2ATK</u>
	27	8	3	0	16	3	0	13

Figure E-6

(7) Milestones. Messages are recorded in a log file at the point in time when an event or aberrant condition occurs. It is useful to have some indication in the message log of the sequencing of these events to place the message in relative perspective. SEAC indicates when various data files have been read and closed, and notes each day change with an entry in the log file. (As shown in Figure E-7, the entry also gives relative computer usage amounts, which are particular to the PRLME minicomputer, on which SEAC was implemented.)

(8) Installation setup. Installations are defined and structured in the GEOLOC File (see Annex C). As in other input routines, installation data are checked for format and validity. The messages in Figure E-8 give several examples of the errors that can occur. Note that when the message appears in the log file, it is accompanied by the record that induced the error.

(9) Monitor file. SEAC is a stand alone model. The execution of the model is directed by the input found on the various data files read during execution. While execution could be viewed from a terminal, the screen output would quickly overwhelm a viewer. SEAC can be interrupted to view the screen, but this halts execution and is a rather inefficient means of checking interim results. It is still useful, however, to be able to check that SEAC is making reasonable progress, thus the reason for the Monitor file shown in Figure E-9. As identifiable milestones are achieved, SEAC records the event in Monitor, along with the cpu time consumed and the simulated time of the action.

(10) ECHO. Sometimes it is necessary to put a computer program "under a microscope." This most often occurs during verification or debugging. For a data-driven program such as SEAC, it is useful to examine program execution relative to the input stream. To allow this, there is a control variable that can be toggled to print both the input record and an indicator of the process or procedure that identified the condition.

b. Data tables. For the most part, SEAC does not place any particular order requirements on the data that are placed in the input files. The model structures the data for efficient internal access. A user can direct SEAC to print out the data tables in the structured order that they are

SEAC DAILY CYCLE CHANGE MESSAGES

```
<<<<<<... NEW DAILY CYCLE AT MODEL TIME 0.00 >>>.
.<<<<DT=      44.70 >>>..<<<<TOT=      174.12 >>>>>>
.
.<<<<<<... NEW DAILY CYCLE AT MODEL TIME 1.00 >>>.
.<<<<DT=      38.41 >>>..<<<<TOT=      212.53 >>>>>>
.
.
.....SEAC TERMINATES AT TIME = 3.00
. DT=      25.92 .. TOT=      269.22 .....
.
.
```

Figure E-7

ERRORS IDENTIFIED DURING GEOLOC FILE PROCESSING

```
### BUILDMAP ERROR...<< BAD COUNTRY SERVICE >>
### BUILDMAP ERROR...<< WRONG CARDTYPE >>
### BUILDMAP ERROR...<< INSTALLATION AMBIGUITY >>
### BUILDMAP ERROR...<< COLLOCATION FAILED >>
### BUILDMAP ERROR...<< COLLOCATED BASE DOESN'T EXIST >>
### BUILDMAP ERROR...<< COMPLEXING >>
### BUILDMAP ERROR...<< MISSING MAPCELL >>
.
.
```

Figure E-8

CONTENTS OF THE SEAC MONITOR FILE

<u>Model Time</u>	<u>Model Action Being Timed</u>	<u>Elapsed CPU Time</u>	<u>Delta Time</u>
0.000	SEAC INITIATED	6.23	0.00
0.000	UOM BUILT	7.75	1.52
0.000	FACTOR BUILT	9.34	1.59
0.000	CATCODE BUILT	26.74	17.40
0.000	ECAPB BUILT	33.66	6.92
0.000	GEOLOC BUILT	76.40	42.74
0.000	DATA INPUT COMPLETED	129.42	53.02
0.000	TIME IN SEAC	174.12	44.70
1.000	TIME IN SEAC	212.53	38.41
2.000	TIME IN SEAC	243.30	30.77
3.000	MODEL FINISHED	269.22	25.92
3.000	REPORTS COMPLETED	315.72	46.50

Figure E-9

maintained within the model. (Note that the examples presented below are in most cases extracts. Many of the actual tables are considerably larger.)

(1) Engineer unit capability. The table shown in Figure E-10 shows engineer capability data as defined by the ECAPB File (see annex C).

(2) Planning factors. The table in Figure E-11 shows the planning factors defined for an assessment. It is a very useful summary, considering the different permissible forms of factors.

(3) CATCODEs and Components. CATCODEs within the model are indexed by CATCODE, service, country, and component type. The input files, however, most likely group CATCODEs by nation and by service. Figure E-12 shows the compact CATCODE-component breakdown that parallels the internal version used by SEAC. The right most column indicates which of the three permissible components are defined for the analysis. The presence or absence of components has particular implications. For example, the absence of an initial standard component is interpreted to mean that there is to be no new construction of facilities of the given CATCODE.

(4) Units of measure. As described in Annex C, the unit of measure (UOM) data compiled for SEAC, and shown in Figure E-13, serve several purposes. First, it establishes what are the valid measurement units for an

ENGINEER CAPABILITY DATA

<u>UTC</u>	<u>MEN</u>	<u>NAT</u>	<u>SVC</u>	<u>HORZ</u>	<u>VERT</u>	<u>GENERAL</u>
48AAA	22	1	1	102.00	17.00	25.00
48AAQ	222	1	1	339.00	534.00	212.00
49GGL	22	1	1	102.00	17.00	25.00
49GH2	67	1	1	97.00	67.00	0.00
49GH3	17	1	1	97.00	67.00	0.00
49GH4	67	1	1	97.00	67.00	0.00
49GH5	32	1	1	97.00	67.00	0.00
49GH6	95	1	1	142.00	262.00	60.00
49GHD	20	1	1	0.00	150.00	0.00
49GHE	32	1	1	45.00	120.00	15.00
49GHF	55	1	1	52.00	240.00	15.00
4AAAD	181	1	1	136.00	704.00	255.00
.						
.						

Figure E-10

SEAC PLANNING FACTORS

<u>FACTOR</u>	<u>NAT</u>	<u>SVC</u>	<u>CATCODE</u>	<u>UNIT AMOUNT</u>	<u>UNIT OF MEASURE</u>
ADA			131E	200.00	SF
			141L	500.00	SF
			721A	400.00	SF
			811A	200.00	KW
BASE	1	1	131A	1600.00	SF
BASE	1	2	116A	803.00	SY
			116B	1600.00	SY
			116C	14900.00	SY
			121B	4.00	OL
			131A	8100.00	SF
			133A	1600.00	SF
			141B	960.00	SF
			141H	2400.00	SF
			141I	2750.00	SF

Figure E-11

CATEGORY CODE/COMPONENT LISTING

CATCODE	FACILITY DESCRIPTION	NAT	SVC	AVAILABLE COMPONENTS	
111A	HARD&AFD PAVE 3IN	1	1		TEMPORARY WAR-DAMAGE
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
111B	HARD&AFD PAVE 3IN	1	1		TEMPORARY WAR-DAMAGE
111C	HELO LANDING PAD	1	1	INITIAL	TEMPORARY WAR-DAMAGE
112A	HELOPAD, TW, PARK AP	1	3	INITIAL	
		1	1		TEMPORARY WAR-DAMAGE
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
113A	HELOPAD, TW, PARK, AP	1	3	INITIAL	
		1	1	INITIAL	TEMPORARY WAR-DAMAGE
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
116A	WASH APRON	1	1	INITIAL	
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
116B	COMP CAL PAD	1	1	INITIAL	TEMPORARY
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
		1	1	INITIAL	
116C	ARM/DISARM PAD	1	1	INITIAL	
		1	2	INITIAL	TEMPORARY WAR-DAMAGE
116D	AFDLPVMT EXPN AM2	1	2	INITIAL	TEMPORARY WAR-DAMAGE

Figure E-12

UNIT OF MEASURE DATA LISTING

Unit of Measure	Unit of Measure (equiv)	Conversion Factor	Damage Criteria
BD	SF	500.000	MAEB
BL	SF	1.000	EMDA
CF	SF	0.125	EMDA
CY	SF	9.000	EMDA
EA	SF	500.000	EMDA
FA	SF	2000.000	MAEB
FB	SF	100.000	EMDA
GA	SF	1.000	EMDA
GM	SF	1.000	MAEB
KG	SF	1.000	MAEB
KV	SF	1.000	MAEF
KW	SF	2.000	MAEF
LF		1.000	EMDL
LN	LF	1.000	EMDL
MB	SF	1.000	MAEF
MI	LF	5280.000	EMDL
OL	SF	100.000	EMDA
OU	SF	500.000	EMDA

Figure E-13

execution of the model. It also associates each UOM with data needed to calculate facility damage.

(5) GEOLOC installations. The definition of theater installations and their function is found in the GEOLOC file. Internal to SEAC is a structure that orders this information by GEOLOC (the input file does not require and in some cases prohibits a strict alphabetical array). The information, such as found in Figure E-14, provides the user a convenient alphabetical listing of installations, installation complexes, and their types.

(6) MAPCELL-GEOLOC mapping. In addition to installation specific data, the GEOLOC File is also the means by which the mapcell-GEOLOC linkage is defined (see Annex D). Since the file will not be sorted in mapcell index order, another convenient listing to assure that data

INSTALLATION GEOLOC AND TYPE ASSIGNMENTS

<u>GEOLOC</u>	<u>INSTALLATION AND TYPE</u>
AACC	SUBCOMPLEX OF GEOLOC EVAT
ALEM	ANSONG IS A OTH
ALVH	SUBCOMPLEX OF GEOLOC ALVP
ALVP	ANYANG RS&D IS A CAMP
CMYG	BROOKLYN IS A ELEC
CRC3	CRC SITE (KNGNG) IS A ELEC
CVBV	PUSAN K-9 IS A AFLD
	PUSAN K-9 IS A AFLD
DJBW	CHANGSAN IS A ELEC
DJDJ	SUBCOMPLEX OF GEOLOC DJBW
DMAK	CHECHON R-605 IS A AFLD
DMEZ	CHEJU-DO TRAINING IS A OTH
DRUD	CHINHAIE R-813 IS A AFLD
DRUF	CHINHAIE IS A PORT
DRUG	SUBCOMPLEX OF GEOLOC DRUF
DRUL	SUBCOMPLEX OF GEOLOC DRUF
DSFG	CHOEJONG SAN SAT TRK IS A ELEC
DSLQ	CHONGJU K-59 IS A AFLD
	CHONGJU K-59 IS A AFLD
DSLJ	CHONGJU (IC) IS A OTH

Figure E-14

MAPCELL-INSTALLATION CROSS-REFERENCE

MAPCELL	GEOLOCS CONTAINED
1	DMEZ
2	
3	QNPW
4	
5	
6	EPW5
7	
8	ZQAA
9	
10	
11	MMFY
12	KAZX
13	
18	
19	
20	
21	
22	HK14 HC07
23	MLWR MLWY NJVE WPK
24	
25	UPUD UPUB
30	USLG EPW1
.	
.	

Figure E-15

assignments have been correctly made is shown in **Figure E-15**. The listing indicates which GEOLOCs have been assigned to specific mapcells.

c. **Process details.** In the paragraph describing SEAC's ECHO option, the need to look closely at how the program is operating was discussed. Two of the most important activities within SEAC are the generation of damage, and the allocation of direct and general engineer support to tasks. To assure that these processes were correctly working, logic was included within the model to report the details of these processes. This ability, however, has since proved useful to investigate why particular results occurred.

(1) **Threat attacks.** The direct calculation of war damage is one of the major innovations in SEAC. Messages are entered in the log file to

indicate when missions were scheduled and what happened to the threat elements on those missions. To see what the results, i.e., damage were, a user can ask SEAC to report on each attack. The threat element, the installation-facility target, and the amount of damage, if any, are provided. Figure E-16 gives

REPORT OF INSTALLATION/FACILITY WAR DAMAGE EVENTS

```

..=====..IL28      2 C HUMPHREYS-
      219A.FAC/DMG. .BLAST. MAEB
            DIST: 33.51, RAD: 31.62
            DIST: 36.07, RAD: 31.62
            DIST: 42.68, RAD: 31.62
            DIST: 29.95, RAD: 31.62  381.1

      131D.FAC/DMG. .BLAST. MAEB
            DIST: 42.64, RAD: 23.40
            DIST: 25.15, RAD: 23.40  122.3
      &HIT& DAMAGE OF 131D  122.3 AT TIME  0.25

..=====..MIG17     2 C HUMPHREYS
      131A.FAC/DMG. .BLAST. MAEB
            DIST: 14.82, RAD: 55.97
            DIST: 14.82, RAD: 55.97  628.3
      &HIT& DAMAGE OF 131A  628.3 AT TIME  0.25

..=====..IL28      1 SEOUL K-16-
      111A .SUR/DAM. &HIT& DAMAGE OF 111A  0.0 AT TIME 0.25
            CRTR  CRTR  CRTR

..=====..MIG17     5 PYONGTAEK A-511/K-6-
      811A .UTL/DMG. .BLAST. MAEF
            DIST: 15.56, RAD: 28.28
            DIST: 39.39, RAD: 28.28  314.16
      &HIT& DAMAGE OF 811A  314.2 AT TIME  0.25

```

Figure E-16

several examples. The first shows the results of an IL-28, Beagle attack against facilities at Camp Humphrey. Since CATCODE 219A is a building (maintenance shop), the evaluation criteria will be the mean area of effectiveness for blast damage (MEAB). "DIST" indicates the distance a bomb falls from the center of a target with radius RAD. The final value shows the total damage, in square feet, to that facility caused by this attack. The

attack against K-16, an airbase, shows that a different procedure is followed when attacking surface targets, i.e. runways, taxiways, etc. For them, damage is measured in craters. Note also that in the final record, the attack against a utility target (an electric power source) results in using the mean area of effectiveness for fragmentation damage.

(2) Engineer resource allocation. SEAC goes to great lengths to emulate engineer support. Engineer units assigned to installations or regions, are analogous to units acting in direct and general support roles. During development of the model, a readout of how capability was being allocated was included to assure correct operation. The messages produced, however, have continued to be useful, especially when a user needs to look closely at why utilization may be low, or that a project might remain in a job queue too long. The messages in Figure E-17 were extracted from one day's results during a test run. The first record shows the result of performing a task that is taken from a job queue at Kunsan. The far right column indicates what engineer capability is found at the installation (or region). The absence of an entry indicates there is no capability at Kunsan; a request then goes to the region (in this case identified as region 72) that provides general support to Kunsan. That region also has no capability, so the request passes on to the next higher region. The engineer capability found in region 7 is more than enough to satisfy the task's needs. The next task also occurs at Kunsan, and follows a similar path to obtaining engineer capability. Note, however, that the engineer pool assets in region 7 have been reduced by the amount of effort used to fulfill the previous task. Another example worth noting is the task arising at Pyongtaek, where there happens to be some engineer capability. It is sufficient to complete the horizontal and vertical portions of the task, but not the general. Note the remaining task requirements that are requested from region 63. Although there were no general hours in Pyongtaek's engineer pool, the model substituted the remaining horizontal and vertical hours as general labor (SEAC has a table for permissible skill substitutions). The region has sufficient general labor in its pool to complete that portion of the task beyond the installation's engineer capability. Note that the next task, at Osan, also requests support from the region, whose available general hours have been reduced by the 178.67 hours applied to the Pyongtaek task.

ENGINEER RESOURCE TASKING RESULTS

Installation/region	Task Requirement			Engr Pool Assets		
	Horz	Vert	Genl	Horz	Vert	Genl
### KUNSAN USAFB K-8	3.35	5.24	2.31			
### CANDO REGION 72	3.35	5.24	2.31			
### CANDO REGION 7	3.35	5.24	2.31	1134.00	1809.90	548.10
### KUNSAN USAFB K-8	64.00	2.00	60.00			
### CANDO REGION 72	64.00	2.00	60.00			
### CANDO REGION 7	64.00	2.00	60.00	1066.64	1802.66	485.79
### C HUMPHREYS	0.64	23.89	24.64			
### CANDO REGION 63	0.64	23.89	24.64	1101.70	1781.04	519.21
### PYONGTAEK A-511/K	10.47	96.34	236.67	12.65	151.99	0.00
### CANDO REGION 63	0.00	0.00	178.67	1101.06	1757.15	494.58
### OSAN K-55	64.00	2.00	60.00			
### CANDO REGION 63	64.00	2.00	60.00	1101.06	1757.15	315.91
### OSAN K-55	64.00	2.00	60.00			
### CANDO REGION 63	64.00	2.00	60.00	1037.06	1755.15	255.91

Figure E-17

d. **Status.** SEAC represents installations, facilities, and engineer units in great detail. Reports on the status of units and installations are useful to analysts who want to see what is happening within SEAC.

(1) **Installation.** Most engineer work occurs at installations. This is because most facilities are located at installations, and engineer work either constructs or repairs facilities. A status report for each installation is routinely produced at the termination of the simulation. The report, an example of which is Figure E-18, gives general information such as the name of the installation, its country and service affiliations, and its current population. It also shows facility status, engineer units assigned to the installations, and the contents of the job queue at the end of the simulation. This report could be produced at any time, and could be useful to a user monitoring requirements and activities at a few installations.

INSTALLATION STATUS REPORT

TVRK ---[PYONGTAEK A-511/K-6/1/1/] STATUS REPORT
 POPULATION: 0, SORTIES: 5, SPF ATTACKS: 0

INSTALLATION ASSETS		REOD	ONHAND	UNDERWAY				
	131A	1600.0	1600.0	0.0				
	442A	0.0	2710.0	0.0				
	721A	0.0	59792.0	0.0				
	722A	0.0	15452.0	0.0				
	730A	0.0	590.0	0.0				
	811A	0.0	1600.0	0.0				
	812A	0.0	46870.0	0.0				
	842A	0.0	15450.0	0.0				
	851A	0.0	0.0	0.0				

		Horz	Vert	Gen	Eff	UTC
ENGR UNITS	2	120.0	742.0	0.0	0.360	4X6AA
		51.7	319.8	0.0	0.360	4X6AA

		Time	Prog/Dur	Size	Pri	Catcode	Type	Reqt
TASKS	2	0.5	3.0/11.0	57.34	1	131A	W	1600.00
		0.5	3.0/ 5.0	343.48	2	811A	W	1600.00

Figure E-18

(2) Engineer units. Another useful report shows the distribution and capability of engineer units across the entire theater. Engineer units are individually identified by their unit type code (UTC) and location -- installation or region. The report also aggregates total unit capability within the location where engineers are assigned. **Figure E-19** gives an example. It shows there are units in subregion 63 and at an installation within the subregion. The subregion entry lists and summarizes the units specifically assigned to it but would not include those units assigned to the installation. This contrasts to capability reports that will be described later in this section. In those reports, the regional capability subsumes all engineer capability assigned to the region, its subregions, and the installations they contain.

e. **Regional reports.** The messages, tables, and displays described up to this point have all reflected input or processing information. The

THEATERWIDE ENGINEER UNIT STATUS REPORT

Region (subregion)		GEOLOC		Horz hrs	Vert hrs	Gen hrs	Rel eff.	UTC		
1	5	AREA	TKDX	1260.0	2011.0	609.0	0.900	4X7ME		
				1134.0	1809.9	548.1				
	51			405.0	555.0	30.0	0.360	4X1AB		
				145.8	199.8	10.8				
	6			1260.0	2011.0	609.0	0.900	4X7ME		
				1134.0	1809.9	548.1				
	62			1260.0	2011.0	609.0	0.900	4X7ME		
				1260.0	2011.0	609.0	0.900	4X7ME		
				2268.0	3619.8	1096.2				
				1260.0	2011.0	609.0	0.900	4X7ME		
	63			51.7	319.8	0.0	0.360	4X6AA		
				1152.6	1925.0	548.1				
				AREA	ALEM	136.0	704.0	255.0	0.270	4WXCG
						223.2	1155.2	418.4	0.270	4WXCG
						223.2	1155.2	418.4	0.270	4WXCG
						223.2	1155.2	418.4	0.270	4WXCG
						217.5	1125.8	407.8		
					TVRK	120.0	742.0	0.0	0.360	4X6AA
						51.7	319.8	0.0	0.360	4X6AA
						61.8	382.2	0.0		
	7					1260.0	2011.0	609.0	0.900	4X7ME
						1134.0	1809.9	548.1		

Figure E-19

regional reports give results which summarize the evaluation of planned engineer support to meet projected workload. To conserve computer time and space, SEAC does not automatically produce reports for all possible areas. Instead, the user designates those regions (see Annex D for discussion of SEAC's regionalization of the theater) for which daily capability, workload, and backlog data are desired. A regional report subsumes the data found in all subregions, mapcells, and installations within the region. Reports also provide other special details such as service, priority, skill, and type of work.

(1) Capability. The daily capability report, shown in Figure E-20, presents two breakouts: the first shows the daily available engineer hours by skill; the second shows those same hours according to service affiliation ("C" := contractor; "A" := Army; etc.).

CAPABILITY REPORT FOR REGIONAL SUMMARY

DAY	HORZ	VERT	OTHER	TOTAL	A	F	M	C	N
0	11088.4	8663.1	19628.6	39380.1	31276.0	8104.2	0.0	0.0	0.0
1	11722.0	8663.1	19628.6	40013.7	31909.6	8104.2	0.0	0.0	0.0
2	12660.4	9366.3	21189.6	43216.3	35112.1	8104.2	0.0	0.0	0.0
3	13810.0	10069.5	22750.6	46630.0	38525.9	8104.2	0.0	0.0	0.0
4	14263.6	10303.9	23270.9	47838.4	39734.2	8104.2	0.0	0.0	0.0
5	14748.4	10772.7	24311.5	49832.6	41728.4	8104.2	0.0	0.0	0.0

Figure E-20

(2) Workload by service. SEAC presently models two types of engineer work: new construction and damage repair. Facility maintenance was not modeled in the initial version of SEAC because of the absence of credible wartime maintenance data. If this data were available, it would be relatively easy to add maintenance to SEAC with its detailed representation of facilities and installations, and the treatment of engineer work as tasks. The workload report, shown in Figure E-21, summarizes when and how much engineer work occurs with each new engineer task that is created. Multiple day projects are distributed over time, rather than indicating their total time occurred on the day the task was created.

(3) Requirements. The requirements report, shown in Figure E-22, actually shows the daily accumulated result of prior days' application of capability to workload. Requirements could just as easily be called cumulative backlog. Each day the task queues found within the boundary of the region are examined and daily hour requirements aggregated in various ways. It does not matter whether the task was placed in the queue that day, or has been sitting in the queue for weeks because of inadequate engineer capability.

(4) Utilization. To assist force designers in assuring that resources are effectively used, SEAC can show the percentage of daily capability that is actually assigned to tasks. Utilization is affected by, among others, location, service, and skill needs. Low utilization may indicate that engineer assets might be better used elsewhere; it might, however, mean that there is a need for horizontal skills, rather than the vertical skills of the units currently available. Utilization reports, such

WORK SUBMISSION REPORT FOR REGIONAL SUMMARY

DAY	A		F		M		C		N	
	NEW CNSTR	WAR DMG	NEW CNSTR	WAR DMG	NEW CNSTR	WAR DMG	NEW CNSTR	WAR DMG	NEW CNSTR	WAR DMG
0	733.4	684.0	696.0	11804.7	0.0	0.0	0.0	0.0	0.0	0.0
1	1104.0	1108.6	1215.6	9914.6	0.0	0.0	0.0	0.0	0.0	0.0
2	10843.5	1363.4	0.0	5645.5	0.0	0.0	0.0	0.0	0.0	0.0
3	3938.4	1353.6	0.0	3343.3	0.0	0.0	0.0	0.0	0.0	0.0
4	3723.4	1078.8	0.0	2290.7	0.0	0.0	0.0	0.0	0.0	0.0
5	3078.4	869.8	0.0	1419.4	0.0	0.0	0.0	0.0	0.0	0.0

Figure E-21

REQUIREMENTS REPORT FOR REGIONAL SUMMARY

DAY	V I T A L			C R I T I C A L			E S S E N T I A L			N E C E S S A R Y		
	HORZ	VERT	OTHER	HORZ	VERT	OTHER	HORZ	VERT	OTHER	HORZ	VERT	OTHER
0	771.2	723.0	259.5	2601.0	1420.6	1249.0	4082.6	2413.6	397.6	0.0	0.0	0.0
1	965.3	1029.1	591.6	1991.9	2332.8	1267.6	4530.0	3600.1	753.9	0.0	0.0	0.0
2	649.3	1113.9	442.9	2126.3	2680.4	1318.1	3580.9	9069.4	4565.8	0.0	0.0	0.0
3	283.7	952.5	303.0	1635.9	2131.7	977.6	3188.7	7890.3	3941.4	0.0	0.0	0.0
4	67.1	458.9	118.1	1438.9	1716.2	723.1	3184.8	7774.3	3903.7	0.0	0.0	0.0
5	6.7	429.6	81.2	1145.6	1459.4	363.1	3164.0	7732.5	3894.0	0.0	0.0	0.0

Figure E-22

as illustrated in Figure E-23, are useful to surface possible maldistributions. There may, however, be reasons for the result ranging from doctrine or support plans, to a need for changes in the model or input files (especially in project input directed by the ORDERS file).

UTILIZATION REPORT FOR REGIONAL SUMMARY

DAY	V I T A L			C R I T I C A L			E S S E N T I A L			N E C E S S A R Y			TOT %
	HORZ	VERT	OTHER	HORZ	VERT	OTHER	HORZ	VERT	OTHER	HORZ	VERT	OTHER	
0	687	557	186	2262	1009	1085	2060	1037	163	0	0	0	23.0
1	618	673	347	272	752	362	1720	865	174	0	0	0	14.5
2	522	819	323	498	1185	507	492	2126	1004	0	0	0	17.3
3	220	687	222	201	905	388	125	1970	1601	0	0	0	13.6
4	63	222	73	299	671	446	144	1939	1609	0	0	0	11.4
5	3	193	36	358	499	210	144	1905	1600	0	0	0	9.9

Figure E-23

(5) In practice. ESC quickly found that these numeric reports, although essential for in-depth analysis, were simply too detailed to use in presenting results to decision makers. Instead, ESC ported the data to graphic terminals that were easily able to plot the data and produce hard copy of those graphs. Another case of a picture being worth a thousand numbers!

LAST PAGE OF ANNEX E